FACULTY OF ENGINEERING
AND ARCHITECTURE

Learning Robotic Cloth Manipulation

Andreas Verleysen

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Francis wyffels, PhD - Prof. Joni Dambre, PhD

Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

May 2022

GHENT
UNIVERSITY

Dankwoord

Dit boek is meer dan de som van gepubliceerde hoofdstukken in wetenschappelijke tijdschriften. Het is de accumulatie van hard werk en volharding, dat niet mogelijk is zonder de ondersteuning van vele anderen. Ik ben een geluksvogel. Een bijzondere groep personen rond mij schrijft mee aan dit verhaal. Ik wens dan ook iedereen die betrokken was bij mijn onderzoek, zowel op professioneel als op persoonlijk vlak, van harte te bedanken. In het bijzonder zij die hier niet bij naam genoemd zijn.

Mijn eerste dankbetuiging gaat uit naar mijn hoofdpromotor professor Francis wyffels. Francis, jouw technische bijdragen aan dit doctoraat zijn vanzelfsprekend. Ik ben je in het bijzonder dankbaar voor jouw toewijding. Je investeert in iedere doctoraatstudent een stukje van jezelf. Jouw hulp gaat veel verder dan wekelijkse feedbackmomenten. Je ging mee op pad om kandidaat-kledijvouwers te ronselen, bent 24/7 bereikbaar, helpt mee de grote lijnen en kleine onderzoekdetails uit te pluizen, biedt oneindig veel kansen en waakt over ons mentaal welzijn. Jouw bezieling straalt af op ons hele team en maakt van het onderzoekslabo een aangename werkomgeving. Voor die werksfeer dien ik ook mijn twee andere AIRO-hoofden te bedanken: mijn copromotor professor Joni Dambre en professor Tony Belpaeme. Mijn waardering gaat ook uit naar alle betrokken juryleden om dit proefschrift kritisch en met volle aandacht onder handen te nemen. Uiteraard kan dit alles niet gerealiseerd worden zonder de nodige financiële en administratieve steun. Hiervoor ben ik dank verschuldigd aan de Vlaamse overheid en aan de administratieve ondersteuning van onze vakgroep en de UGent.

De achtste verdieping in iGent, de AIRO onderzoeksgroep, is al vijf jaar mijn tweede thuis. Er leeft een dynamische en intellectueel uitdagende sfeer, die te danken is aan de alumni en aan de huidige collega's: Lio, Dries, Alexander, Gabriel, Jonas, Ira, Jeroen, Fréderic, Luthffi, Tom, Olivier, Matthijs, Victor, Thomas, Remko, Dries, Matthias, Mathieu, Maxime, Pieter, Rembert, Natacha, Peter, Zimcke, Axel, Benedikt, Bjarne, Jeanne, Maria, Qiaoqiao, Ruben, Stefan en Tanguy.

Tom, zonder jou had mijn eerste jaar in het robotlabo een eenzaam bestaan geweest. Olivier, jouw gekke LaTeX tovenarij bespaarde me vele uren tijdens het schrijven van dit doctoraat. Onze gezellige babbels waren een welgekomen afwisseling. Matthijs, jouw implementatiehulp en denkwerk waren onmisbaar voor enkele spannende resultaten in dit boek. Victor, het is steeds een uitdaging om onze brainstorms tot een uur te beperken. Thomas, Remko en Dries, ik kijk al uit naar jullie verwezenlijkingen.

De technische en wetenschappelijke hulp van mijn collega's was bijzonder waardevol, net zoals de morele ondersteuning van mijn vrienden. Wieland, Jeroen, Thomas, Tim en Rik, we kennen elkaar al sinds de kleuterklas. Bedankt voor jullie vriendschap. Jaarlijks een week gamen in de Ardennen als detox is iets wat ik aan weinig mensen uitgelegd krijg. Maura en Tom, ik kijk iedere maand opnieuw uit naar onze kookpartijen en boardgameavonden. Sam, jij bracht me in 2013 in contact met de groep waarin ik nu dit doctoraat afrond. Bedankt. De *band*: Jeroen, Lieven, Renaat, maar toch ook nog een beetje Laurens. We zijn gekend onder drie namen (momenteel als *Anna Lies*, zoek ons op Spotify!), maar we zijn toch al dertien jaar dezelfde individuen en samen vormen jullie mijn grootste muzikale uitlaatklep.

Ik wil verder ook mijn ouders, mijn zus Anastasia en mijn broer David bedanken voor het warme nest waarin ik opgroeide. Mama en papa, bedankt voor alle onvoorwaardelijke steun. Ik kon zorgeloos studeren en mezelf ontwikkelen tot de persoon die ik nu ben. Het is niet vanzelfsprekend, maar ik heb het toch altijd zo mogen ervaren. Qliff en Storm, jullie zijn de liefste en meest knuffelbare collega's tijdens het vele thuiswerken door de COVID-19-pandemie.

Tot slot, Stefanie, jij staat al zoveel jaar naast mijn zijde. Je was erbij toen we van de middelbare schoolbanken kwamen, we hebben samen gestudeerd, begonnen op hetzelfde moment te werken en je steunde me voluit toen ik de stap naar de academische wereld maakte. En je bent er nog steeds. Je geeft me een reden om 's avonds het computerscherm uit te zetten en naar buiten te gaan in onze tuin, samen te koken, onze honden te knuffelen en de wereld rond te reizen. Je maakt van ons huis een thuis. Bedankt voor alles.

Andreas Verleysen, May 25, 2022

Summary

Endowing robots with dexterous manipulation skills could spur economic welfare, create leisure time in society, reduce harmful manual labour, and provide care for an ageing population. However, while robots are producing our cars, we are still left to our own devices for doing the laundry at home. This shortcoming is due to the major difficulties in perceiving and handling the variability the real world presents. Robots in modern manufacturing require engineers to produce a safe and predictable environment: objects arrive at the same location, in the same orientation, and the robot is preprogrammed to perform a specific manipulation. Unfortunately, the need for a predictable environment is undesirable in dynamic environments that handle a wide range of objects, often in the presence of human activity. For example, should a human worker first unfold all clothes such that a robot can easily find the corner points and perform folding? Indeed, the high variability in modern production environments and households requires robots to handle objects that can take arbitrary shapes, weights, and configurations. This diversity renders traditional robotic control algorithms and grippers unsuitable for deployment in dynamic environments.

To find methods that can handle the ever-changing nature of human environments, we study the perception and manipulation of objects that provide an infinite amount of variations: deformable objects. A deformable object changes shape on force interaction. Deformable objects are omnipresent in industry and society: food, paper, clothes, fruit, cables and sutures, among

others. In particular, we study the task of automating the folding of clothes. Folding clothes is a common household task that will potentially be performed by service robots in the future. Handling cloth is also relevant in manufacturing, where technical textile is processed, and in the fashion industry.

Dealing with the deformable nature of textiles requires fundamental improvements in both hardware and software. Mechanical engineering needs to incorporate actuators, links, joints and sensors into the limited space of a hand while using soft materials similar to the human skin. In addition to engineering more capable hands, control algorithms need to loosen assumptions about the environment in which robots operate. It is unattainable to expect highly deformable objects like cloth to always be in the same configuration before manipulating them with a robot.

A solution for dealing with real-world variability can be found in the machine learning field. In particular, deep RL combines the function approximation capabilities of deep neural networks with the learn by trial-and-error formalism provided by RL. Deep RL has shown to be capable of driving cars, flying helicopters and manipulating rigid objects. However, the data requirements for training highly parameterized functions, like neural networks, are considerable. This data-hungry property causes an incongruity between the representational learning features of deep neural networks and the high cost of generating real robotic trials.

Our research focuses on reducing the required learning data for systems that perceive and manipulate clothing items. We implement a cloth simulation method to generate synthetic data, utilize smart textiles for state estimation of cloth, crowdsource a dataset of people folding clothing, and propose a method to estimate how well people are folding clothing without providing labels.

Actuating a physical robot is slow, expensive and potentially dangerous. For this reason, roboticists resort to physics simulators that simulate the robot's and environment's dynamics. However, *there exists no integrated robot and cloth simulator for use in learning experiments.* Cloth simulators are built for offline render farms in the film industry, or for the game industry that sacrifices fidelity for real-time rendering. Unfortunately, cloth simulation for robotic learning requires performance characteristics similar to online rendering and accuracy aspects found in offline rendering. For this reason, we implement a custom cloth dynamics simulation on GPU and integrate it in the robotic simulation functionality of the Unity game engine. We found that we can utilize deep RL to train an agent in our simulation to fold a rectangular piece of cloth twice within 24 hours of wall time on standard computational hardware.

The developed cloth simulation assumes full accessibility to the state of the cloth. However, *state estimation of cloth in the real world relies on complex vision-based pipelines or high-cost sensing technology.* We avoid this complexity and cost by integrating inexpensive tactile sensing technology into a cloth. The cloth becomes an active smart cloth by training a classifier that uses the tactile sensing data to estimate its state. We use this smart cloth to train a low-cost robotic platform to fold the cloth using RL. Our results demonstrate that it is possible to develop a smart cloth with off-the-shelf components and use it effectively for training on a real robotic platform.

Our smart cloth bridges the gap between our cloth simulation on GPU and state estimation in the real world. However, it is still required to distil a scalar value that indicates task progress in order to acquire manipulation skills using RL. We believe that learning the reward function from demonstrations may overcome human bias in reward engineering. Unfortunately, when starting our research *there existed no large dataset with people folding clothing.* We fill this gap by crowdsourcing a dataset of people folding clothes. Our dataset consists of roughly 300.000 multiperspective RGB-D frames, annotated with pose trajectories, quality labels and timestamps indicating substeps. This dataset can be used to benchmark research in action recognition and bootstrap learning by using example demonstrations.

Learning from demonstrations is a prevalent domain in the robotics learning community. However, using our cloth folding dataset requires mapping the movements of demonstrators to the embodiment of a robot. Additionally, behavioural cloning is prone to blindly imitating trajectories instead of understanding how actions relate to solving the task. For this reason, estimating how well a process is being executed is preferred to learning the policy from demonstrations. Unfortunately, existing methods couple the learning of rewards with policy learning, thereby inheriting all problems associated with RL. To decouple reward and policy learning, we propose a method to learn the task progression from multiperspective videos of example demonstrations. We avoid incorporating human bias in the labelling process by using time as a self-supervised signal for learning. We demonstrate the first results on expressing task progression of people folding clothing, without labelling any data.

General-purpose robots are not yet among us. Robots that are capable of working in dynamic environments will require a holistic view of software and hardware. We demonstrated the benefits of this approach by outsourcing the intelligence for state estimation to the cloth instead of the robot. By developing a smart cloth, we trained a robot to fold cloth in-vivo within a day. Extrapolating an integrated approach on hardware and software leads to *embodied intelligence* in which morphology closes the loop with control: co-optimizing the body and brain will allow evolving manipulators tailored to the tasks, and use them to build a representation of how the world works. Robots can use that feedback to understand how actions influence the environment and learn to solve tasks by using human examples, instrumented objects and their own experiences. This holistic process will enable future robots to understand human intent and solve a large repertoire of manipulation tasks.

Samenvatting

Robots met gelijkaardige manipulatievaardigheden als de mens zullen in de toekomst economische welvaart bevorderen, tijd vrijmaken in de samenleving, gevaarlijke handenarbeid overnemen en zorg voorzien aan een vergrijzende bevolking. Terwijl robots onze auto's produceren, zijn we helaas nog steeds op onszelf aangewezen om thuis de was te doen. Deze tekortkoming is te wijten aan de moeilijkheden die we ondervinden bij het omgaan met de variabiliteit die te vinden is in de echte wereld. Robots die aanwezig zijn in moderne productievestigingen vereisen van ingenieurs dat zij een veilige en voorspelbare omgeving voorzien. Een robot wordt namelijk geprogrammeerd om specifieke bewegingen uit te voeren. Daardoor wordt er verwacht dat objecten steeds op dezelfde locatie en op dezelfde manier aangeleverd worden. Helaas is het inrichten van een voorspelbare en gestructureerde omgeving ongewenst in dynamische omgevingen waar een breed scala aan objecten wordt verwerkt, vaak in de aanwezigheid van menselijke activiteit. Bijvoorbeeld, moet een menselijke arbeider in een textielbedrijf eerst alle kleren ontvouwen, zodat een robot gemakkelijk de hoekpunten kan vinden en de kledij kan vouwen? De grote variabiliteit in moderne productieomgevingen en huishoudens vereist immers dat robots voorwerpen verwerken die willekeurige vormen, gewichten en configuraties kunnen aannemen. Deze diversiteit maakt traditionele robotbesturingsalgoritmen en robotgrijpers onbruikbaar om in te zetten in dynamische omgevingen.

Om methodes te ontwikkelen die kunnen omgaan met de hoog veranderlijke aard van menselijke omgevingen, bestuderen we

de perceptie en manipulatie van objecten die een oneindige hoeveelheid variaties bieden: vervormbare objecten. Een vervormbaar object of voorwerp verandert van vorm als er een kracht wordt uitgeoefend. Vervormbare objecten zijn alomtegenwoordig in de industrie en de maatschappij: onder andere voedsel, papier, kledij, fruit, kabels en hechtingen. We bestuderen de taak van het automatiseren van het vouwen van kledij. Het vouwen van kledij is een veelvoorkomende huishoudelijke taak die in de toekomst mogelijk door dienstrobots kan worden uitgevoerd. Het manipuleren van kledij is ook relevant in de productie-industrie, waar technisch textiel wordt verwerkt, en in de mode-industrie.

Het omgaan met de vervormbare aard van textiel vereist vooruitgang in zowel hardware als software. Op mechanisch vlak moeten actuatoren, scharnieren, sensoren en andere onderdelen ingebouwd worden in de beperkte ruimte van een hand. Daarbij dienen ze zachte materialen te gebruiken die gelijkaardige eigenschappen delen met de menselijke huid. Naast de ontwikkeling van meer capabele handen moeten de besturingsalgoritmen minder assumpties maken over de omgeving waarin robots werken. Het is bijvoorbeeld onrealistisch om te verwachten dat sterk vervormbare voorwerpen zoals textiel zich altijd in dezelfde toestand bevinden om ze door een robot te laten manipuleren.

Omgaan met de variabiliteit in de echte wereld kan gebeuren aan de hand van machinaal leren. Meer concreet is er diep versterkingsgebaseerd leren (RL) dat de functiebenaderingsmogelijkheden van diepe neurale netwerken combineert met het trial-and-error formalisme van RL. RL wordt reeds gebruikt om auto's te besturen, met helikopters te vliegen en niet vervormbare voorwerpen te manipuleren. Helaas hebben neurale netwerken heel veel data nodig om patronen te leren herkennen. Deze hoeveelheid aan data is vaak te kostelijk om met een echte robot te genereren.

Het onderzoek in dit boek behandelt het reduceren van de dataset benodigdheden om systemen te trainen die kledij kunnen herkennen en manipuleren. We implementeren een kledijsimulatie om synthetische data te genereren, bouwen een slim textiel dat kan vertellen of het al dan niet gevouwen is, verzamelen een dataset van mensen die kledij plooien, en stellen een systeem voor dat autonoom leert te vertellen hoe goed mensen kledij aan het vouwen zijn.

Een robot aansturen is duur, traag en mogelijks gevaarlijk. Robotici gebruiken daardoor simulatieomgevingen die het gedrag van de robot en de omgeving simuleren. *Er bestaat echter geen robot simulator die compatibel is met bestaande kledijsimulaties.* Textielsimulatie wordt gebruikt in de filmindustrie, waar offline berekeningen worden uitgevoerd op vele machines. Er bestaan ook textielsimulaties in videospellen. Deze zijn echter van lage kwaliteit opdat ze real-time kunnen uitgerekend worden. Vanwege het ontbreken van een geschikte kledijsimulatie, schrijven we onze eigen kledijsimulator en integreren we deze in een bestaande robotsimulator. Onze experimenten tonen aan dat we onze simulator kunnen gebruiken om een robot kledij te leren vouwen, en dit binnen 24 uur rekentijd op gewone computationele hardware.

Tijdens het autonoom leren met onze kledij- of textielsimulator veronderstellen we dat onze robot de volledige toestand van het textiel kan opvragen. *De toestand van textiel berekenen vraagt echter complexe berekeningen of dure sensortechnologie.* Om deze complexiteit te vermijden, instrumenteren we het textiel door goedkope voelsensoren te integreren. We maken het textiel *slim* door methodes uit het machinaal leren te gebruiken die op basis van de sensordata kunnen vertellen of het doek al dan niet gevouwen is. Vervolgens gebruiken we het slimme textiel om een robot te leren de was te plooien. Onze resultaten tonen aan dat het gebruik van een slim textiel toestaat om op een goedkoop, fysisch robotplatform te leren vouwen.

Het slimme textiel overbrugt de kloof tussen onze simulatieomgeving en de werkelijke wereld. Het is echter nog steeds nodig om een scalaire waarde uit te rekenen die de taakvooruitgang aangeeft tijdens de uitvoering ervan. We verwachten dat het beter is om de taakvooruitgang te leren dan ze manueel te specifiëren. Het manueel specifiëren kan zorgen voor menselijke vertekening. Er bestaat echter nog geen grote dataset van mensen die kledij vouwen. Omwille van deze reden verzamelen we onze eigen dataset. Onze dataset bevat ruwweg 300.000 RGB-D beelden die geannoteerd zijn met de positie van de armen, kwaliteitslabels en tijdstempels die de deelstappen aanduiden. Deze dataset kan worden gebruikt voor onderzoek naar actieherkenning en het versnellen van leren met gebruik van voorbeelden.

Het is vereist om de bewegingen van de menselijke voorbeelden in onze dataset te vertalen naar de morfologie van de robot.

Om te vermijden dat deze vertaalslag leidt tot het blind na-apen van bewegingen, geven we er de voorkeur aan om een beloningssignaal te leren in plaats van de handelingen zelf. Er bestaat echter nog geen methode die een beloningssignaal kan leren, althans niet zonder simultaan ook het gewenste gedrag te leren. Deze koppeling zorgt ervoor dat alle problemen die geassocieerd worden met versterkingsgebaseerd leren worden overgenomen bij het leren van beloningen. We introduceren een methode die het leren van beloningen en gedrag ontkoppelt door rechtstreeks het taakverloop te voorspellen op basis van camerabeelden van menselijke voorbeelden. Aangezien onze methode niet vereist om individuele beelden te voorzien van een vooruitgangsgetal, vermijden we het insijpelen van menselijke vertekening. Onze methode is het eerste systeem dat taakvooruitgang van mensen die kledij vouwen uitdrukt in een prestatiemaatstaf.

De robotbutler is nog niet voor morgen. Robots moeten eerst leren werken in dynamische omgevingen. Hiervoor moet toekomstig onderzoek zich toespitsen op het integreren van software en hardware. In ons onderzoek hebben we de kracht hiervan aangetoond door middel van een slim textiel dat de robot vertelt hoe het gevouwen is. Zo vermijden we dat de robot zelf de toestand van het textiel moet leren. Door deze eigenschap kan de robot binnen de dag leren om het textiel te vouwen. Het verder doordrijven van deze geïntegreerde visie op hardware en software leidt tot intelligentie die ingebouwd is in het lichaam zelf. Het simultaan optimaliseren van het lichaam en het brein staat toe om robotgrijpers te ontwikkelen die aangepast zijn om specifieke taken uit te voeren. Robots kunnen deze grijpers gebruiken om representaties te leren die toestaan om te begrijpen hoe de wereld werkt. Zo zullen robots de gevolgen van hun acties leren begrijpen en uiteindelijk taken leren uitvoeren door te interageren met hun omgeving, te kijken naar hoe mensen de taak uitvoeren en door feedback van geïnstrumenteerde objecten op te vragen. Dit holistisch proces zal de toekomstige robotbutler toestaan om mensen te begrijpen en een breed gamma aan taken te leren uitvoeren.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolution Neural Network
DOF	Degree Of Freedom
DQN	Deep Q-Network
FEM	Finite Element Methods
GAN	Generative Adversarial Network
GPU	Graphical Processing Unit
HMM	Hidden Markov Model
MDP	Markov Decision Process
ML	Machine Learning
NFQ	Neural Fitted Q-iteration
POMDP	partially observable Markov decision process
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
SVM	Support Vector Machine
t-SNE	T-Distributed Stochastic Neighbor Embedding
TCN	Time Contrastive Network

TD	Temporal-Difference
UMAP	Uniform Manifold Approximation and Projection

1

1 Introduction

Robotic manipulation of clothing presents challenges we need to solve in order to build general-purpose robots.

1 Introduction

The research field of robotics promises to relieve humanity from repetitive tasks. Yet, why is there currently no robot that ties our shoelaces, weeds our gardens, and folds our shirts? These tasks break the boundaries of the scripted environment for which existing robot technologies are programmed. Programmed robots follow routines to assemble cars and weld miniaturized electrical components at astounding speeds and with fine-grained precision. However, these robots operate in a cage where every instruction is precisely defined with minimal room for deviation. Robots that do operate outside a safety cage are persistently supervised by humans. For example, the da Vinci® robot allows performing surgery by teleoperating robotic arms through a hand-operated console. Bringing this robotic automation to our daily lives will significantly impact industry and society. Robots will sort and pack all types of objects in warehouses, install electrical wiring in our cars, help elderly people dress, and clean up our houses.

For the robot butler to become a reality, we need to endow robots with dexterous manipulation skills. Similar to how you effortlessly turn the page of this book by grasping and reorienting the corner of the page, robot hands require dexterity to perform the same impressive repertoire of tasks as the human hand can. This revolution for robotic hand dexterity will occur at the frontiers of both hardware and software. First, mechanical engineering needs to incorporate actuators, links, joints and sensors into the limited space of a hand while using soft materials similar to the human skin. Second, control algorithms need to loosen assumptions about the environment in which robots operate. Nowadays, robotic manipulation solutions require environments to be fully specified and the object's configuration to be fixed. However, the real world presents an infinite supply of configurations. To find solutions for this problem, we study the manipulation of objects that provide an infinite amount of configurations: deformable objects.

1.1 Deformable object manipulation and robotic laundry

Many everyday objects deform upon force interaction: the wires we use to charge our phones, the clothes we carry, and the sutures doctors use for stitching wounds. In this research, we consider the problem of **automating the task of folding clothing**. Folding clothing is part of the laundry cycle, visualized in Figure 1.1, and a potential task for future household robots. With ageing populations, increasing quality of life and importance of work-life balance, automating dull and repetitive tasks in households merits benefits from a social and financial point of view. Outsourcing tasks to robots allows freeing up time or budget if a domestic helper is employed. For older people or people with limited mobility, domestic task automation can give a feeling of independence as the need for the assistance of home care workers is reduced. Additionally, there is a significant portion of the industry working with garments, textiles and other deformable objects that could also benefit from developments in general-purpose robotic automation.

Although automating domestic tasks offers a sound business case, developments in automation have primarily focussed on industrial automation. This is because industrial tasks are much simpler to automate compared to domestic environments where it is hard to exert exact control and enforce necessary prerequisites. Even in research, folding clothing has received little attention. The main reason is that robotics research focuses primarily on grasping and manipulating rigid objects. Rigid objects do not deform during interaction and significantly reduce the required information. For example, grasping your morning coffee cup requires only knowing the position of the

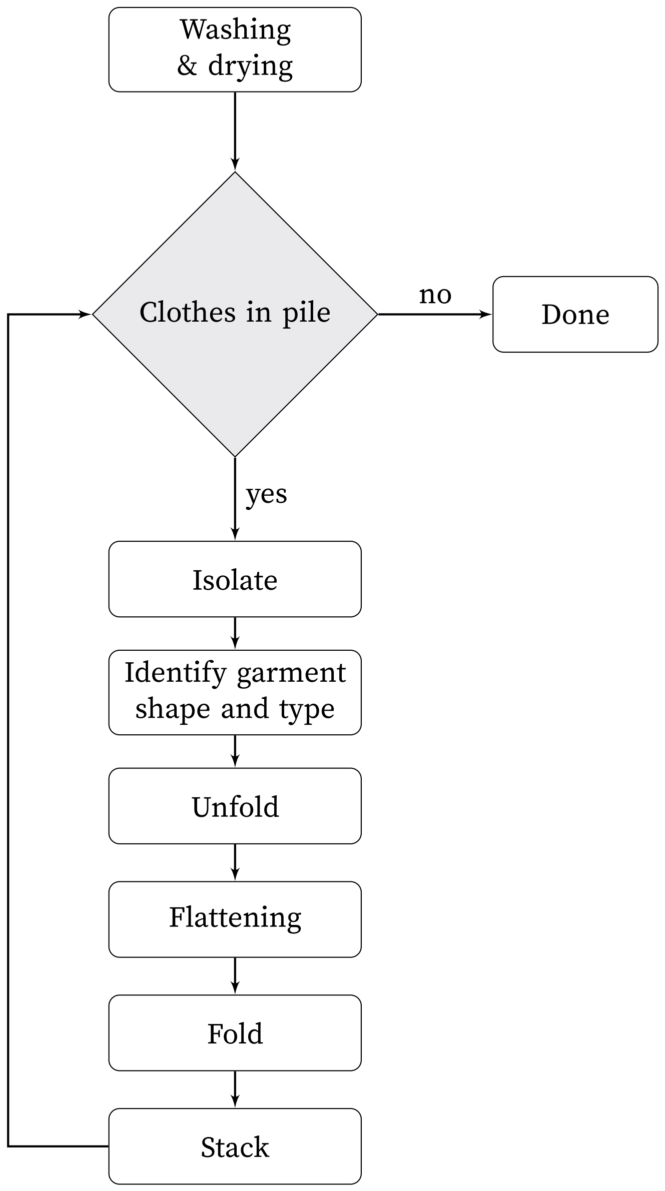


Figure 1.1 Task flow for robotic laundry. Figure adapted from (Hamajima and Kakikura 1996).

cup's handle. On the contrary, if the handle is highly deformable, you also need to reason about its shape and how it will deform as a result of the planned manipulation. Unfortunately, planning for deformations is not a part of traditional robotic control pipelines.

1.2 Traditional control pipelines do not work for folding clothes

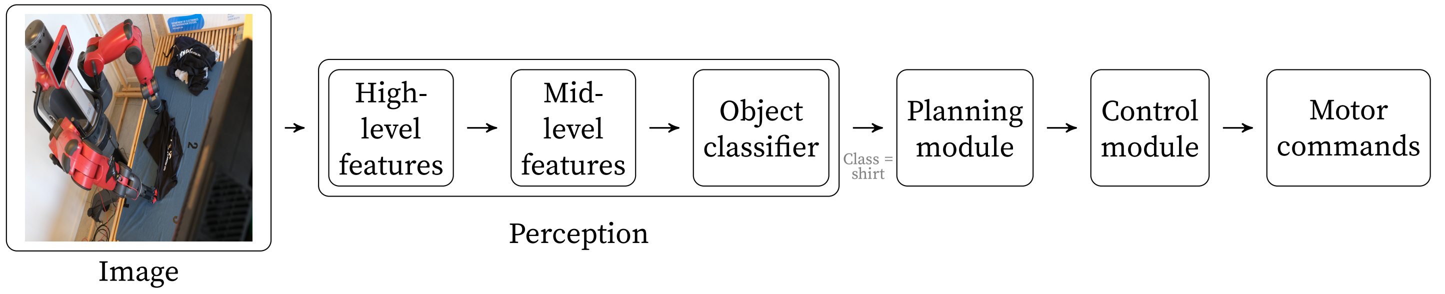
A daily-life example found in common kitchens illustrates the workings of today's industrial robots. Instead of manually slicing food into smaller pieces for cooking, one can use a food processor to automate the slicing process. Using the food processor requires structuring the environment: one needs to equip the correct blade on the motor shaft, preprocess the food into a manageable size for the blade and feed it through the tube while activating the motor. Deviating from this setting can lead to minor failures and even harmful situations. For example, inserting oversized food slices might lead to motor stalling. More dangerously, there is no intelligence in consumer-grade appliances stopping the motor from turning the blades if you decide to insert your hand into the feeding tube. Luckily, a manufacturer can easily prevent such dangerous operations through the mechanical design of the tube.

Current robotic control pipelines are organized similarly to the given food processor example: structuring the environment and decomposing the problem. Decomposing a large problem into subproblems is a general problem-solving and engineering paradigm that allows making assumptions that simplify the solution strategy. In robotics, this modular approach is omnipresent and has led to incredible levels of automation and an increase in productivity (Graetz and Michaels 2018). For example, the pick-and-place contest in the Amazon Robotics Challenges provided the participants with prior knowledge about the objects that need to be grasped[1] and allowed them to

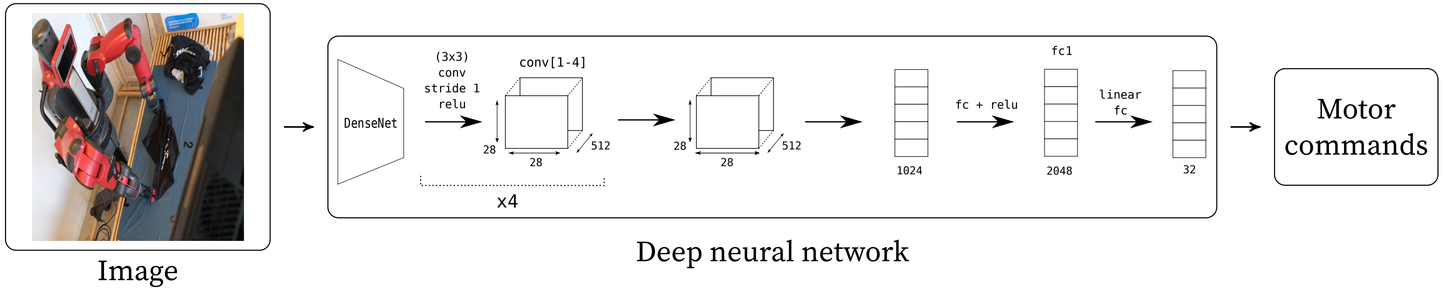
1. In the last edition of 2017, this assumption was scaled back as the teams knew 50 % of the items only 45 minutes beforehand.

design their own storage system. Winning solutions (Eppner et al. 2017; Morrison et al. 2018) solve this structured problem by breaking it down into simpler problems, solved by individual modules for perception, planning and control. These modules are then factorized into systems performing tasks like labelling pixels to objects, segmenting the target object, and fitting it with bounding boxes. However, fully modularized architectures that do not adapt to deviations and failures elsewhere in the control pipeline will repeatedly fail when variations occur in the routine. Some of the most advanced robotics research and development teams demonstrate this in the 2015 DARPA Robotics Challenge Finals (DARPA 2015). The consequences of exposing robots to unstructured environments can be viewed in published video clips (Spectrum 2015): million-dollar robots tumbling to the ground in, frankly, hilarious ways. In one example, a robot is supposedly turning a valve. However, the robot's walking path is misaligned, making it stand next to the valve instead of in front of it. Consequently, the robot is performing a rotating movement with its arm in thin air while assuming resistance of the valve for balancing. The missing counterbalance causes the robot to fall on the ground. The rotating valve failure brings the inner working of control loops to light: a sequence of isolated modules that solve decomposed tasks with insufficient regard for failures down- or upstream in the pipeline.

Robots that manipulate deformable objects employ similar pipelines: a sequence of modules with functionalities ranging from garment type identification to corner point detection and executing preprogrammed folding sequences. We visualize a canonical pipeline for folding clothing in Figure 1.2a. These approaches are assumption-heavy and can have limited generalization. Engineered pipelines also have difficulty scaling: successful implementations require 24 minutes to fold a single shirt (Maitin-Shepard et al. 2010). This motivates our research to look in the direction of *employing learning-based methods* to create autonomous systems that can perform robust grasp synthesis when faced with the high amount of variations that occur in the real world.



(a) Canonical robotic control pipeline for folding clothing. Each module solves an isolated task and passes the result to the next module.



(b) End-to-end robotic control pipeline for folding clothing. An input image is passed to a black-box deep neural network which sends direct motor commands.

Figure 1.2 Standard robotic control pipelines versus end-to-end architectures.

1.3 From engineered pipelines to end-to-end learning

Machine learning is a subset of AI methods that can learn patterns from examples. The successes of deep reinforcement learning, a machine learning method that learns by trial-and-error, in playing Atari video games (Mnih et al. 2015) and beating human champions at the game of Go (Silver et al. 2016), has driven robotics researchers to adopt machine learning across the whole pipeline. This led to an end-to-end learning approach in which the robot has to figure out how to move its arms using a single camera image. In the example pipeline of Figure 1.2a, every functional module is integrally replaced with a deep neural network, as shown in Figure 1.2b.

Unfortunately, end-to-end learning for robotics is privileged to organizations that contain massive computational power and expensive robotic farms. The robots in these farms operate *interactively* in an environment: they perform actions by trial-and-error to maximize predefined success criteria. Although robots learning from scratch yield entertaining footage, similar to the falling robots of DARPA 2015, it tires quickly as learning from scratch on a real robot takes tremendous amounts of time and is potentially unsafe. From the perspective of how humans learn, it makes no sense to have a system learn from scratch. Imagine asking a human baby to do the laundry while it still has to learn how to move its limbs. An additional limitation is that many robotic systems only use visual inputs. The popularity of this approach can be explained by the accessibility of cameras and the successes of deep learning in computer vision. Yet, interacting with the environment requires fusing multiple senses such as vision and touch. Hence, if we want physical systems to solve tasks in dynamic environments, they require some form of prior knowledge and multimodal inputs to accelerate learning.

For cloth folding robots, we have, on the one hand, robots that use engineered pipelines and take 24 minutes to fold a shirt (Maitin-Shepard et al. 2010). On the other hand, there exist robots that learn to fold from scratch but require over 100.000 physical robot-environment interactions (Matas, James, and Davison 2018). Our research explores solutions in the middle of the modelling and learning paradigm.

1.4 Accelerating learning of robotic manipulation of deformable objects

Learning is considered to be a central component for autonomous systems to deal with real-world variability (Kroemer, Niekum, and Konidaris 2021). However, learning-based methods generally require many examples and interactions with the environment. For example, Levine et al. (2018) record 800.000 grasping attempts using 12 robots to learn to grasp rigid objects. This dataset requirement is problematic for robots that are expensive to operate. Supervising multiple robots simultaneously for millions of interactions with cloth is an unpractical and unscalable solution. For real robots to learn to fold clothing articles, there is a need to accelerate the learning process. In this dissertation, we research the following four topics to accelerate learning: (i) generation of cloth folding datasets for learning; (ii) using simulations as a safe and fast environment for learning; (iii) facilitating state estimation of cloth; and (iv) understanding task intent rather than copying human behaviour. We discuss these topics in the following sections.

1.4.1 Datasets

A critical piece of the puzzle for building intelligent systems comes in the form of *data*. The standard paradigm in machine learning is to provide a dataset with, for example, images of folded and unfolded shirts and train a model that learns to distinguish between the two. When learning how to solve tasks, datasets in the form of task demonstrations have been used for training autonomous driving cars (Bojarski et al. 2016), learning quadruped robotic locomotion (Peng et al. 2020), drone flight (Giusti et al. 2016) and learning to place dishes in a plate rack (Finn, Levine, and Abbeel 2016). However, in the field of learning robotic manipulation skills for clothing, no dataset with example demonstrations of people folding clothing exist. The lack of folding datasets can be attributed to the domain being less studied, the difficulty in collecting high-quality, diverse samples and the high state space nature of deformable objects, making it hard to generalize example demonstrations to unseen examples. We hypothesize that collecting a dataset of people folding clothing can be used for bootstrapping learning of manipulation skills. This dataset should be collected "in the wild" to avoid biases and promote diversity. At the same time, the data collection should be standardized in order to facilitate ad-hoc labelling and ensure data quality.

1.4.2 Simulation

Whereas a real-world dataset can be expensive to obtain, synthetic approaches allow the generation of massive datasets and quick experimentation. Hypotheses in robotics are often first tested in a virtual environment to allow fast prototyping and validation. To utilize simulation for robotic learning, realistic modelling of the dynamic behaviour of robots and cloth is required. The development of robot simulation technologies has come a great way since the demonstration of deep learning methods for robotics. However, cloth simulation has primarily been developed for offline simulation used in the movie industry or real-time simulations in games where fidelity for cloth is secondary to game-play simulation. Cloth simulation for robotic learning needs to be fast, but not real-time as is the case in games, and requires realistic dynamics. Realistic robot and cloth simulation allow fast experimentation and facilitate transferring the results from simulation to the real world. We believe that integrating physically plausible cloth simulation with robotic simulators enables to learn cloth manipulation skills in simulation with transfer to the real world.

1.4.3 Instrumentation

When we use our hands to crush a plastic cup, multiple sensors of our body activate: our eyes observe the deformations, our ears register the amplitude of the impact, our hands notify us of how much force we are applying, and our proprioceptive system signals us to which extent our fingers are closed. This rich interplay of multiple modalities in the human cognitive system is in stark contrast to robotic manipulation pipelines that are primarily vision-based. Vision is important for robotic manipulation: it helps to infer the object location relative to the robot end-effectors, it helps to understand the object's geometry and some of its physical properties. Furthermore, commercial cameras are readily available and accessible compared to other sensors such as tactile sensors. Nevertheless, considering the giant leaps of object recognition with deep learning, robots still struggle to recognize objects in more difficult contexts such as partial occlusion, transparent object and moving objects (Guo et al. 2014; Sajjan et al. 2019; Ojha and Sakhare 2015). Incorporating heterogeneous sources of information can alleviate problems when state estimation cannot be directly observed from pixels. For example, finding the occluded corner of a crumbled towel with tactile sensing simplifies the folding process considerably. We denote the process of adding sensory information to the learning environment of a robot as **instrumentation**. The goal is to redirect the large focus on using vision-based state estimation towards the application of other sensor modalities in the environment such as tactile sensing in a cloth and force sensing in the fingers. Our hypothesis is that some modalities encode parts of the state in a more compact way than vision. This semantically more meaningful encoding accelerates learning, which is important in robotics where real rollouts are expensive.

1.4.4 Understanding task intent

The human ability to infer task intent from demonstration is in stark contrast with imitation learning approaches from the machine learning field. Imitation learning for autonomous cars, for example, answers the question "with how many degrees would a human driver turn the steering wheel now that this tree is in front of me?". Although the correct answer is probably to avoid coming into the situation that a tree is in front of the car, no human driver would likely be able to provide an exact answer.

Instead of creating a dataset of task scenarios with associated human actions, the interactive trial-and-error approach of RL shares more semantics with learning from a human perspective. For example, humans encourage desirable behaviour in dogs by means of treats or punishment. In this example, the snack is the reward signal. Applying the idea of reward and punishment to robotics is a popular learning approach but requires finding the appropriate reward signal. In the case of cloth folding, it is ambiguous to write down a mathematical formula that describes how well a person is folding a clothing article. How does one quantify the number of wrinkles to a performance measure? Does it matter that a folded towel does not contain perfectly straight lines? What is the definition of a folded shirt? Finding a reward signal requires identifying, measuring and quantifying all aspects of performance. This might not be possible for all problems, like cloth folding.

Compared to dogs, infants are known to understand task intent behind human behaviour without requiring explicit reward signals (Warneken and Tomasello 2006). We argue that a similar approach can be applied for the case of robotic folding and learning in general. By learning the reward function from human demonstrations, without labelling, one can use this reward signal in an RL setting. This idea is generally known as inverse RL. However, inverse RL learns a reward function in conjunction with learning how to behave. We propose decoupling this process and learning a reward function separately to speed-up training and use it for other applications like process monitoring.

1.5 Research outline

Our main goal is to investigate solutions to accelerate learning-based approaches for learning to fold clothes with robots. We focus on the domains described in the previous Section 1.4: generation of real datasets, the use of simulation technologies, instrumentation and learning a reward function for cloth folding. Our main contributions can be listed as follows:

- Collecting a crowdsourced dataset of people folding clothing articles.
- Estimating the state of cloth with smart textile and proving its utility in a real-world, low-cost robotic setup.
- Proposing and validating a methodology to learn a reward function without providing labels.

To structure the content of our research, we first provide an overview of the field of robotic manipulation of deformable objects in Chapter 2. We provide the introduction and state-of-the-art in a tutorial and survey style chapter. As a first experiment, we study the use of robot and cloth simulation for fast experimentation in Chapter 3. Next, we deal with a shortcoming when transferring from simulation to the real world: the unavailability of a full state specification of the cloth. We measure the state of the cloth by developing a smart textile in Chapter 4. This smart cloth consists of tactile sensors that are used in a machine learning model to communicate whether the cloth is crumbled, folded or unfolded. We demonstrate the effectiveness of the smart cloth on a real robotic setup that learns to fold cloth from scratch using RL. To speed up this approach, we collect a crowdsourced dataset of people folding clothing in Chapter 5. We use this dataset in Chapter 6 in which we learn a reward function that expresses how well people are folding clothing in this dataset. Notable in our approach is that no progress labels are required. In the following Chapter 7, we zoom out to take a birds-eye perspective on the field of robotic cloth folding in order to highlight future areas of improvement, based on our experience and the research done in our work. Finally, we summarize our work and provide concluding remarks in Chapter 8.

1.6 Publications

All chapters in this book are based on work published during the course of this research. The exceptions are the introduction and conclusion of this book. The work described in Chapter 3 is not published as similar research was already being published and the goal of our research is not to provide state-of-the-art technology for cloth simulation. We provide a list of publications below:

- Chapter 2 and Chapter 7:Andreas Verleysen and Francis wyffels. 2022 (expected). “Learning to fold cloth: a survey.” *The International Journal of Robotics Research,* Submitted, under review
- Chapter 4: Andreas Verleysen, Thomas Holvoet, Remko Proesmans, Cedric Den Haese, and Francis wyffels. 2020. “Simpler learning of robotic manipulation of clothing by utilizing DIY smart textile technology.” *APPLIED SCIENCES-BASEL* 10 (12): 10. ISSN: 2076-3417. http://dx.doi.org/10.3390/app10124088
- Based on Chapter 4: Remko Proesmans, Andreas Verleysen, Robbe Vleugels, Paula Veske, Victor-Louis De Gusseme, and Francis Wyffels. 2022. “Modular piezoresistive smart textile for state estimation of cloths.” *Sensors* 22 (1). ISSN: 1424-8220. https://doi.org/10.3390/s22010222. https://www.mdpi.com/1424-8220/22/1/222
- Chapter 5: Andreas Verleysen, Matthijs Biondina, and Francis wyffels. 2020. “Video dataset of human demonstrations of folding clothing for robotic folding.” *The International Journal of Robotics Research,* ISSN: 0278-3649. http://dx.doi.org/10.1177/0278364920940408
- Chapter 6: Andreas Verleysen, Matthijs Biondina, and Francis wyffels. 2022. “Learning self-supervised task progression metrics: a case of cloth folding.” *Applied Intelligence,* Accepted, https://doi.org/10.1007/s10489-022-03466-8

2

2 Background and review of related work

This chapter provides a comprehensive background on robotic manipulation of deformable objects in a survey style.

2 Background and review of related work

The following chapter provides the preliminaries and a review of relevant work in the field of robotic manipulation of deformable objects. First, we introduce motor control architectures for rigid body manipulation and review how they differ from deformable object manipulation in Section 2.1. To give historical context, we then discuss how *standard robotic manipulation pipelines* can be used for manipulating deformable objects in Section 2.2. Next, we discuss how the inherent limitations of engineered motor control architectures can be overcome by using *learning-based methods*. We distinguish between learning with (Section 2.3) and without (Section 2.4) examples, and from interaction with the environment (Section 2.5). Critical to robotic learning of manipulation skills is some metric of task success, generally labelled as the reward function. The role and methods to obtain *reward functions* for robotic learning, and deformable objects manipulation in particular, is reviewed in Section 2.5.2. Given the general property that learning-based methods are data-hungry, we continue this discussion by reviewing the role of *large datasets* for robotic learning in Section 2.6. An alternative approach to generating data is to use synthetic data. To this end, we discuss the role of *simulation* and the corresponding transferability problems in Section 2.7. Finally, we discuss the idea and related literature of *instrumenting the process with sensors* to facilitate the learning process in the manipulation environment in Section 2.8.

2.1 Manipulating deformable objects

This section provides important historical context and prior work in the rigid and deformable object manipulation literature. First, we discuss the traditional control approach to rigid object manipulation for robots and how these methods are challenging to generalize towards deformable object manipulation. Next, we provide a definition and categorization of deformable objects. For each category, we provide common tasks and solutions identified in literature.

2.1.1 Manipulating rigid objects

Grasping and manipulation problems in robotics are traditionally solved by manually engineering subsystems for perception, planning and control (Siciliano, Khatib, and Kröger 2008). A popular approach is using images as input observation to control the robot's motions. This approach is motivated by the advantage that images enable closed-loop control: non-contact and real-time measurements of the environment can be used to provide feedback to the motion trajectory of the robot. This principle is generally known as visual servoing (Hutchinson, Hager, and Corke 1996) and was first introduced in 1979 by Hill (1979). An archetypical pipeline consists of the following steps to grasp and manipulate an object (Corke et al. 1996). First, observations such as images are used to estimate the state of the object. This state estimation stage usually executes pixel manipulations and image filtering in order to extract features. This object state is used to interpret the scene to calculate the relative position of the target object from the robot end-effector. Once the object is identified, it can be modelled to identify a suitable grasping point. Next, these grasping points are given to a motion planning system that calculates a trajectory to move the end-effector to the desired position and orientation. Finally, a low-level controller sends motor commands to the actuators to move the robot. An example of this archetype control pipeline is displayed in Figure 2.1.

Engineering modular, hand-tuned motor control pipelines have been successful for applications in manufacturing (Clocksin

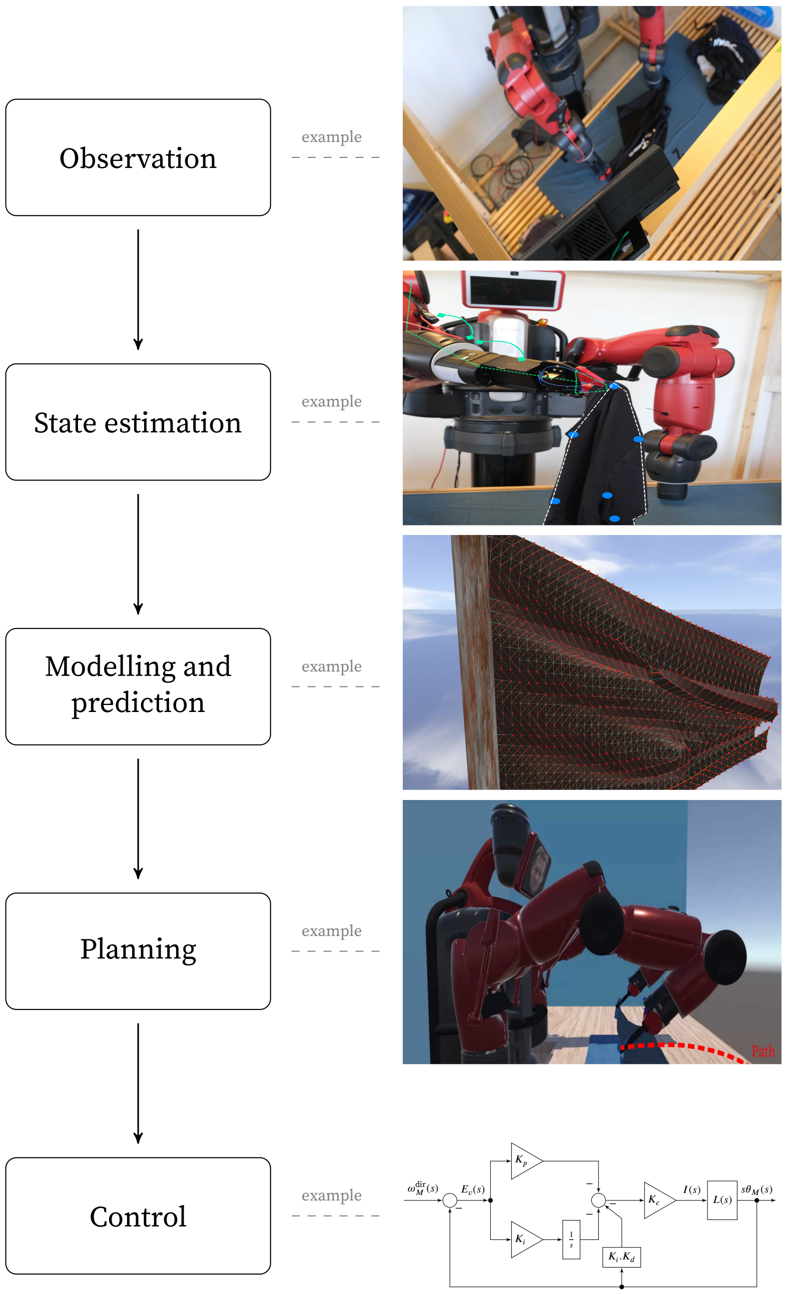
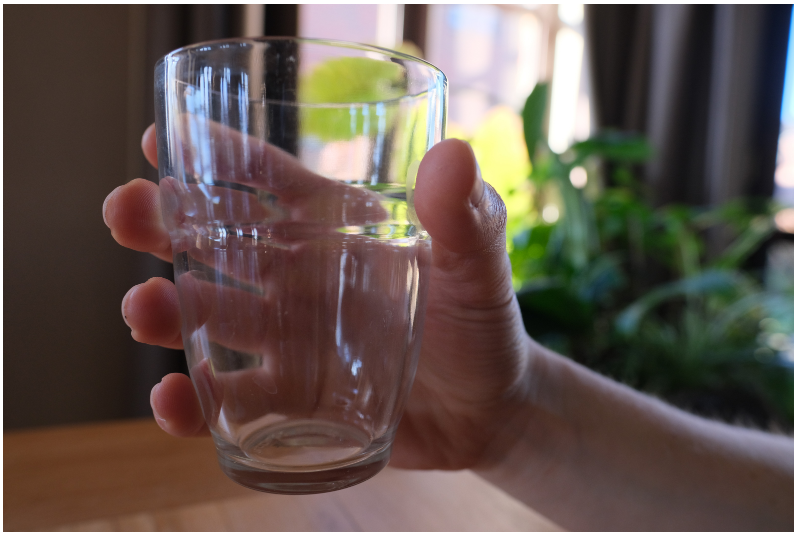


Figure 2.1 Canonical engineered manipulation pipeline with examples of each module. Cameras record observations that are used to estimate the state of the cloth downstream. The modelling module calculates the deformations on the cloth if certain manipulations are executed. A planning module calculates the desired end-effector trajectory and sends the corresponding joint position to a low-level controller.

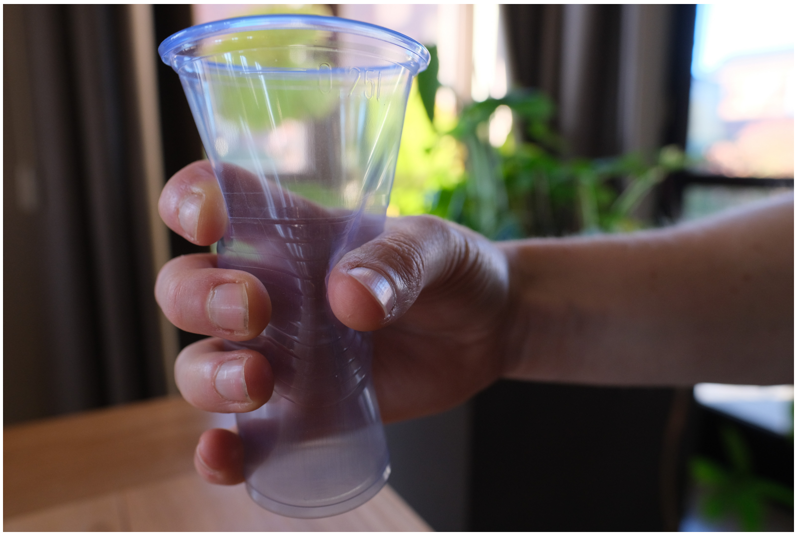
et al. 1985; Mochizuki, Takahashi, and Hata 1987), car steering (Dickmanns and Graefe 1988), robotic ping-pong (Andersson 1987), juggling (Rizzi and Koditschek 1993) as well as fruit picking (Harrell, Slaughter, and Adsit 1989). However, all of these applications operate under the condition of rigid objects: the shape of the object will not change on contact. When manipulating objects, this is of importance for determining stable grasping points. More concretely, restraining rigid objects relies on *form closure* (Nguyen 1988) or *force closure* (Bicchi 1995): fully constraining relative motion of the object or having contact points that can counteract an external wrench through friction. However, in the case of deformable objects, the object can deform during grasping and manipulation. This leads to exponentially higher dimensional configuration spaces compared to rigid object manipulation (Foresti and Pellegrino 2004). For example, achieving form closure becomes impossible as it requires immobilizing every degree of freedom. Similarly, force closure becomes computationally intractable as it requires constantly incorporating the adapted shape of the object. For example, we visually show (Figure 2.2) the deformations that occur when grasping a plastic cup versus a rigid glass when trying to achieve force closure. Furthermore, manipulation requires reasoning about the target shape of the object. These properties make many rigid object manipulation techniques hard to extend in the deformable object domain. Unfortunately, to date, the vast majority of robotic manipulation work deals with rigid objects whereas many objects are of deformable nature (Siciliano, Khatib, and Kröger 2008).

2.1.2 Deformable objects: definition, categorization, tasks and solutions

A deformable object is an object whose shape changes when being subject to an external force. This deformation can be temporary and reversible (*elastic*), permanent (*plastic*) or a combination of both (*elasto-plastic*). Deformable objects are found in industrial settings, agriculture and household items. A common categorization (Saadat and Nan 2002; Jiménez 2012) is based on the geometry of the object: how many dimensions are significantly



(a)



(b)

Figure 2.2 Force closure on a rigid glass (a) and a deformable, plastic cup (b). The deformations of the plastic cup need to be taken into account when calculating a grasping pose.

larger than the other dimensions. The rationale for this categorization is given by small dimensions of the object having a negligible impact on the deformable properties. A canonical example where this property is applied is found in the sheet metal bending industry: the thickness of metal sheets is neglected when computing the required manipulations for bending (Duflou, Váncza, and Aerens 2005). A consequence of this categorization is that 3D objects can be considered as 2D deformable objects. For example, both a hollow rubber ball and plush ball can be considered 3D deformable objects based on their dimensionality. However, when considering the dimensions that only impacts the deformable properties, a hollow ball is 2D deformable object as the thickness can be neglected for manipulation.

Deformable objects

A deformable object is an object whose shape changes on interaction and can be categorized based on the number of dimensions negligible for manipulation planning.

In its simplest setting, the deformable object is one dimensional: ropes, strings, cables, threads and catheters, among others. Some of these examples are shown in Figure 2.3. These objects are also known as deformable linear objects. The term *linear* refers to one dimension being dominant over the other two dimensions. Common tasks for deformable linear objects involve grasping and manipulating ropes, for example, knot tying. Early motor control architectures for solving tasks regarding deformable linear objects used either an open-loop approach or simple visual servoing to execute the motion. An early work clearly demonstrating modular control pipelines is the project of Inaba and Inoue in 1987. Their method employs visual servoing for manipulating a rope into a ring and then tying the rope. Their perception module uses stereo images to detect the rope and the ring. The planning module is hard-coded to iterate through a set of predefined steps while using the detected centre of the ring and 3D coordinates of the endpoints of the rope from the perception module. An inverse kinematics module provides a target trajectory to the low-level controller. Similar modular pipelines can be found in (Remde, Henrich, and Wörn

1999) for grasping a rope and in (Saha and Isto 2007) where knots are tied with needles using probabilistic trajectories of the rope from a simulated model. Incorporating motion primitives, i.e. a predefined set of motor actions corresponding to high-level actions, in the planning module is used in (Yamakawa et al. 2008; Vinh et al. 2012) to tie knots in a rope.

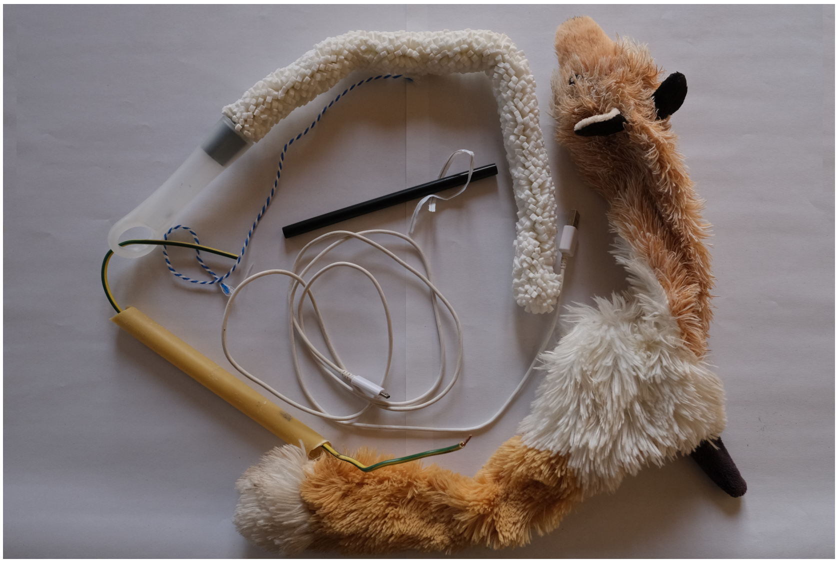


Figure 2.3 Examples of deformable linear objects and an application of putting an electrical wire into a rigid pipe.

Deformable *linear* objects become deformable *planar* objects when two dimensions are significantly larger than the third dimension. In this case, the planning module can disregard the thickness of the material for manipulation. Canonical examples are given in Figure 2.4 and contain objects such as clothing, thin-shelled objects like plastic bottles, fabric, paper, and plastic bags, deformable sheets, cards and foam materials. A classic example of paper folding is robotic origami, in which a robot has to sculpt a piece of paper into the desired shape by folding. This problem was tackled with an open-loop control architecture in (Balkcom and Mason 2008) to produce a folded hat. Elbrechter, Haschke, and Ritter (2012) takes this a step further by using vision, simulation and fiducial markers on the paper to grasp and fold a paper with a five-fingered end-effector.

Related to origami is carton folding and metal sheet bending. Typical control strategies (Lu and Akella 1999; Liu and Dai 2003; Aomura and Koguchi 2002) consist of finding the correct locations and sequence of bending operations by modelling the object as a collection of panes articulated through hinge joints. Robotic manipulation of bags has been less studied due to the complexity of modelling and manipulating bags. To circumvent this complexity, dedicated hardware has been researched for grasping (Kazerooni and Foley 2005) and unloading (Kirchheim, Burwinkel, and Echelmeyer 2008) sacks. A general-purpose two-fingered robotic gripper is used in (Klingbeil et al. 2011) to grasp objects from a table, search the barcode and drop the object into a bag. The planner uses 3D points clouds of depth images taken by a camera. However, they assume the bag is already open for insertion and do not consider any possible deformations caused by touching or dropping an item into the bag.

In the context of this research, it is of interest to note that garments satisfy the same geometrical property of having one negligible dimension as objects such as paper and plastic bottles. However, the main characteristic distinguishing cloth is the compression strength: compared to other two-dimensional deformable objects, cloth does not possess any significant compression strength. Given that the current work deals with manipulations of clothing items, we dedicate Section 2.2 to elaborate on cloth manipulation pipelines.

The final category of deformable objects is *volumetric deformable objects* whose deformations across all dimensions of the object are of relevance. Some examples are shown in Figure 2.5: objects such as food, plush toys and sponges. In the case of food products, deformations can be caused by both grasping and processing operations such as slicing. In general, 3D deformable objects are the least researched type of deformable objects (Sanchez et al. 2018). An exception to this is soft tissue, which is important for medical application. We refer the reader to the review paper by Taylor et al. (2016) for an overview of medical robots in surgery applications. An overview of robotic manipulation of food products is given in Chua, Ilschner, and Caldwell (2003).

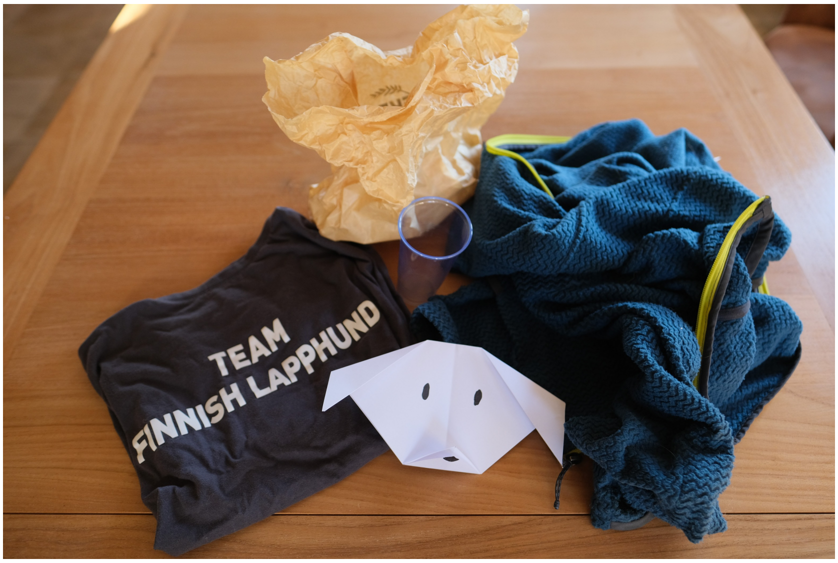


Figure 2.4 Examples of 2D deformable objects: origami, paper bag, shirt, jumper and a cup.

2.2 Engineered cloth folding pipelines

A high-level categorization of cloth tasks are *sensing* of material properties, *grasping* a single clothing item in a cluttered environment and task-specific *manipulation* applications. Specific applications dealing with manipulation of cloth are among others: folding clothing items, hanging cloth on a rod and bedsheet folding. In this dissertation, we focus on the application of cloth folding. A complete cloth folding pipeline is visualized in Figure 2.6. This folding pipeline typically consists of the following subtasks: (1) grasping an isolated garment, (2) bringing it into a folded configuration and (3) stacking it on top of other folded garments. The second step in this process is often subdivided into unfolding, flattening and folding. Most of the work in robotic cloth folding deals with a single subtask instead of providing solutions to the complete pipeline. Two notable exceptions that consider the whole robotic folding pipeline is the work of Doumanoglou et al. (2016) and Maitin-Shepard et al. (2010), which is discussed at the end of this section.

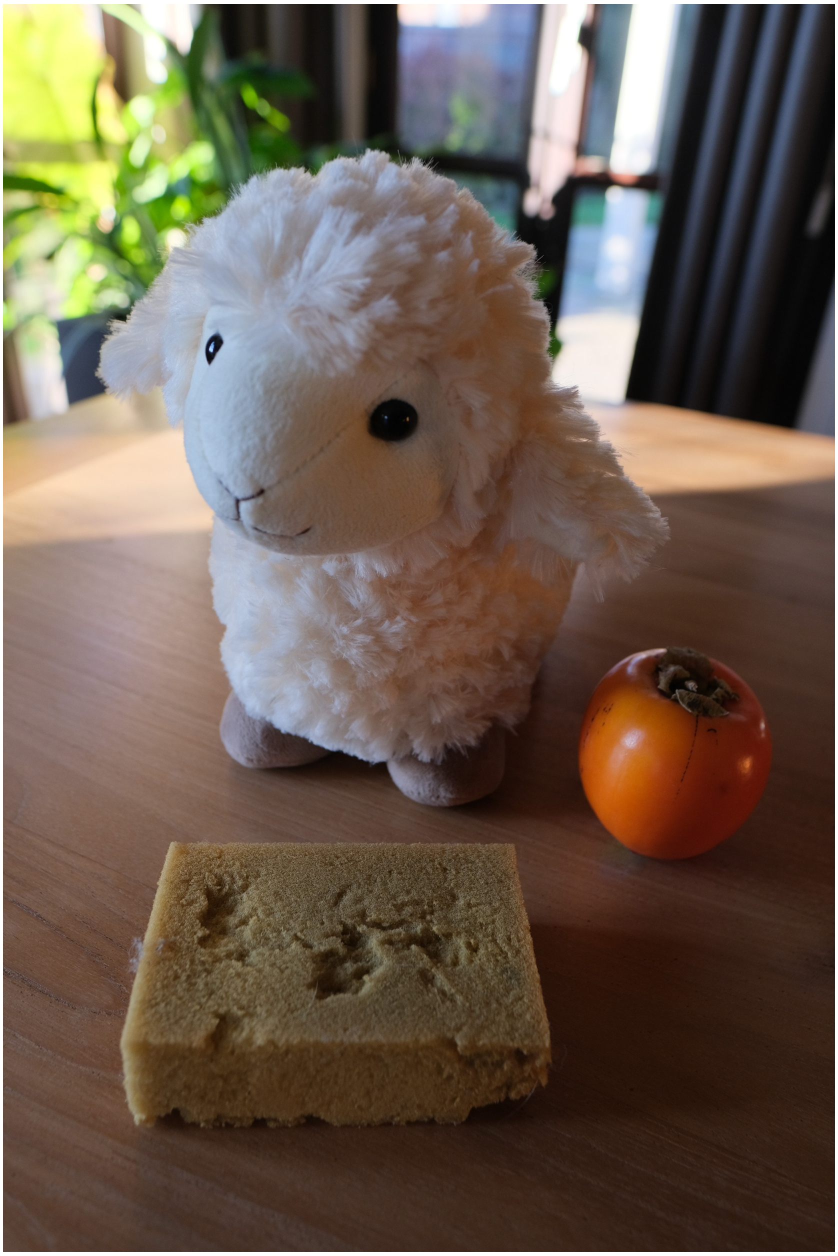


Figure 2.5 Examples of volumetric deformable objects: sponge, fruit and plush toy

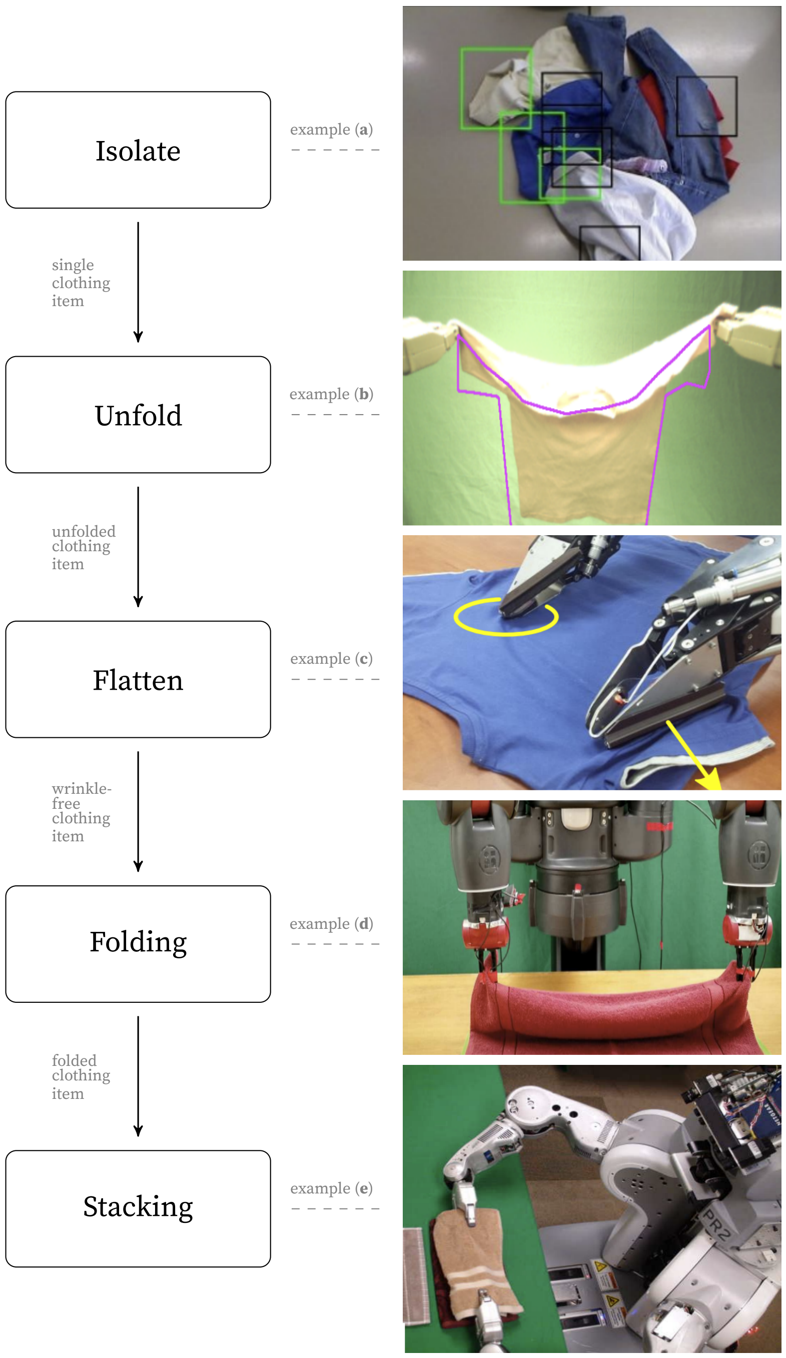


Figure 2.6 Cloth folding pipeline for robotic manipulation with sub-tasks. Sources are the following: (**a**): (Ramisa et al. 2012), (**b**): (Cusumano-Towner et al. 2011), (**c**): (Doumanoglou et al. 2016), (**d**): (Yinxiao Li et al. 2015), (**e**): (Maitin-Shepard et al. 2010)

An obvious solution for cloth folding is to use specialized hardware in a constrained environment. Gomesh et al. (2013) propose an actuated flipfold[1] that automates folding of shirts. More complex commercially available products exist such as the FoldiMate®[2]. However, such products do not generalize towards general cloth folding, do not leverage general-purpose robotic hardware and have proven difficult to bring commercially available[3]. This is why most research considers the use of general-purpose robot arms, possibly with dedicated gripper and instrumentation, as elaborated in Section 2.8.

Much of the literature around cloth folding deals with solving subtasks of the folding pipeline. In the following subsections, we summarise important work concerning solving each of the subtasks. An exhaustive overview of methods on how to solve particular subtasks, focusing on grasping type and manipulation primitives, in the cloth folding pipeline is given (Borràs, Alenyà, and Torras 2020a). Finally, we describe two important works that provide an integrated system for solving the complete folding pipeline.

2.2.1 Grasping

Grasping a piece of cloth requires isolating a single piece of garment from a pile of clothing articles and making sure that one functional piece is grasped. In (Ramisa et al. 2012), this is done by grasping shirts via the collar. Visual servoing is used with preprocessed features on depth data. Their method achieves a grasping success rate of 70%. However, the performance drops to 30% when other types of garments are present on the table. Monsó, Alenyà, and Torras (2012) separate all clothing articles with a robot manipulator. Their work employs a POMDP to model

1. A flipfold is a device consisting of four panels joined by hinges. The four panels lineup with the two sleeves, top-centre and bottom-centre of the shirt respectively. The hinges allow the panels to rotate inwards. This movement takes the cloth with it and as such makes the folds.

2. https://foldimate.com/

3. FoldiMate has been in prototype development for nine years at the time of writing.

the uncertainty in the state estimation caused by the occlusion of clothing articles.

2.2.2 Pose estimation

After grasping a clothing article, pose estimation is usually done such that the type and configuration of the cloth can be brought into an unfolded state, ready for folding. Garment pose estimation has been done by matching video images to simulation models (Kita and Kita 2002), using machine learning models (Li, Chen, and Allen 2014; Yinxiao Li et al. 2014) or instrumentation via fiducial markers (Bersch, Pitzer, and Kammel 2011).

2.2.3 Unfolding

Unfolding is an important step as it allows to bring the article into a known configuration from which predefined folding strategies can be employed. A general approach to unfold clothing articles is to exploit gravity; by grasping the article at strategic points, gravity will remove arbitrary folds. Hamajima and Kakikura (1998) exploit this gravitational trick by regrasping the hemlines of the garments. The hemlines are detected using the shadows and shape of the cloth. Cusumano-Towner et al. (2011) unfolds shirts and trousers in a two-staged pipeline using HMMs, a cloth simulator and a planning algorithm. By inputting the clothing article type, size and grasping points for the gripper to the HMM, it can estimate the garment's configuration. This configuration is used by the simulation model to find the minimum-energy configuration. This is the configuration in which the garments triangulated mesh vertices have minimum gravitational potential energy. Then, the planning module repeatedly executes trajectories to regrasp the clothing article until it is in a known configuration. Then, the planning module brings the garment into the hard-coded, unfolded configuration. Their method achieves a 66% success rate. Doumanoglou et al. (2014) solves the same task but reduces the number of software modules by repeatedly regrasping the lowest hanging point of the garment. This brings the clothing item into a known configuration. Next, the robot unfolds the article by searching two grasping points using a POMDP.

2.2.4 Flattening

Before folding the cloth into the desired configuration, it is necessary to remove wrinkles caused by unfolding the cloth. Moreover, folding often relies on template matching, which is made more difficult when there are wrinkles present. A dedicated method for cloth flattening is proposed in (L. Sun et al. 2015). Their method assumes the clothing item is unfolded on a table. They employ RGB-D data to find wrinkles and represent them as fifth-order polynomials. The largest wrinkle is then flattened by using preprogrammed motions of the arms. In (Bryan Willimon, Stan Birchfield, and Ian Walker 2011), a washcloth is flattened in two phases. In the first phase, they iteratively pull the cloth away from or towards its centroid to remove minor wrinkles. The second phase utilizes depth information to determine regions of interest with a high degree of wrinkles and the necessary direction for removing them.

2.2.5 Folding

Bersch, Pitzer, and Kammel (2011) execute an open-loop motor control trajectory to fold clothing after unfolding it using fiducial markers on the cloth. The folding loop exhibits a common human strategy to fold cloth: grasp the garment by the shoulders, rotate the sleeves inwards and fold the shirt inwards while placing it on the table. In (Berg et al. 2010), they employ a geometry-based folding method that folds over predefined lines. Their method relies on using gravity to immobilize parts of the garment such that parts of the cloth become rigid objects. (Miller et al. 2012) estimate the pose of the garment by fitting a user-specified polygon representation to the detected cloth contours. Then, they apply the same method as (Berg et al. 2010) to fold the cloth. (Yamakawa, Namiki, and Ishikawa 2011) start folding cloth in midair, held by its corners, with an algebraic representation of the cloth. They use this simulation model to estimate the pose of the end-effectors at exact intervals such that the open-loop trajectory of the points describes a folded garment. Contrary to previously mentioned approaches which rely on a dual-robot arm platform, (Petrík et al. 2017) considers folding with a single robotic arm. They compute a trajectory in simulation based on the grasping location and the folding line of the garment. However, the literature concerning folding with a single robotic arm is rather scarce due to its limited applications.

2.2.6 Full pipeline

The first example of a complete cloth folding pipeline is the work of Maitin-Shepard et al. (2010). The task starts from an unorganized pile of crumbled towels and ends when all articles are stacked in a folded configuration on top of each other. Given that the end of their pipeline consists of executing predefined trajectories, much of their method relies on predecessor steps to bring the clothing into an exactly known configuration. Their method starts with colour segmentation on the image to select the central clothing article. Next, the grasped towel is rotated and regrasped in order to find and grasp the corners visually. Unfolding is done by shaking, twisting and pulling the towel taut. Finally, they run a predefined, open-loop trajectory to lay down the unfolded towel on the table and fold it. The pipeline takes 24 minutes to execute, with the grasp point detection phase being the largest bottleneck.

The second full robotic cloth folding implementation is engineered by Doumanoglou et al. (2016). Their setup considers folding a pile of shirts, towels and trousers. By segmenting an image from the pile on colour, an isolated piece is grasped. By repeatedly regrasping the lowest hanging point of the garment, they reduce the amount of possible cloth configuration to classify the garment shape using random forests (Breiman 2001). The unfolding procedure is equivalent to (Maitin-Shepard et al. 2010) and requires a garment classifier, grasp point detector, and a pose estimation module. Flattening the shirt is done using a brush tool on a dedicated cloth folding gripper. Wrinkles are detected by comparing the contours with existing polygonal models of flattened cloths. Finally, the fold is executed using polygon-matching methods the contours to predefined templates. Their system achieves a throughput of six minutes per garment with a 79% success rate. The slowest step in their pipeline is detecting the desired grasping points for unfolding.

As a concluding remark of this section, we observe a reoccurring theme: a divide-and-conquer methodology leads to a loss of information between the different stages, resulting in the accumulation of errors. For example, Doumanoglou et al. (2016) report difficulties when folding towels because the perception system labels them as shirts. These individual components are built in a laboratory environment with certain assumptions which are likely to be violated in an unstructured, complex environment. Together with inaccurate sensor readings, the deformation of the robot's links deteriorates these systems' accuracy. In order for robots to be useful in unstructured environments with complex dynamics, there is a need for controllers that are able to perform robust grasp synthesis when faced with unseen conditions.

2.3 Learning robotic manipulation tasks with labels

Machine learning, a domain of artificial intelligence, is the study of algorithms that allow computers to learn from and make predictions based on data. Machine learning provides a way to deal with the inherent systematic and random errors in robotic systems and variability in unstructured environments. This is because learning-based methods optimize for the grasping task, which implicitly adapts the behaviour to imperfections in the system, such as inaccurate sensor readings.

The following sections provide a short review of the fundamentals of relevant machine learning techniques. We discuss learning with and without labels and by trial-and-error. We discuss their relevant applications in robotic manipulation with a focus on the manipulation of deformable objects.

2.3.1 Supervised learning

Supervised learning is a machine learning paradigm that operates under the setting where there is a set of *input* variables, for example image pixels, that exert influence over other *output* variables, for example whether there is a shirt or trouser in the image.

Supervised learning

Supervised learning involves learning a mapping from input data $\mathbf{x}$ to the output data y, provided by the supervisor.

The components building up a machine learning system are the dataset, the model, loss function and optimization algorithm. These components are discussed next. Formally, we can denote the input data as a set $\mathscr{X}$ consisting of vector $\mathbf{x}^{(i)} \in \mathscr{X}$ with the superscript i referring to the ith observation. In the machine learning domain, this set of predictor variables is often called *features*. The set $\mathscr{Y}$ contains the output variables $y^{(i)} \in \mathscr{Y}$. Concatenating tuples of $\left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right), i \in 1, \ldots, N \right\}$, often called *examples*, leads to a dataset which can be used for learning. Central in this learning procedure is the idea of *function approximation* in which a function f, parametrized by $\boldsymbol{\theta}$, maps an input $\mathbf{x}^{(i)}$ to its corresponding output $y^{(i)}$:

$$f(\mathbf{x}; \boldsymbol{\theta}) : \mathscr{X} \mapsto \mathscr{Y}.$$

This mapping, also called *model* or *hypothesis*, comes in many forms such as linear models, tree-based methods, support vector machines and neural networks[4]. The goal of the learning procedure then becomes to adjust the parameters $\boldsymbol{\theta}$ of the model such that a certain performance measure $\mathscr{P}$ is optimized. This metric, called *loss function* $\mathscr{L}$ in machine learning jargon, is specific to the task and domain in which the learning is taking place. In robotic folding for example, the robot might be presented with a candidate grasping pose $\mathbf{u}$. The robot then has to predict the probability $\hat{y} = Q_{\boldsymbol{\theta}}(\mathbf{u}, \mathbf{x}) = \mathbb{E}\left[S | \mathbf{u}, \mathbf{x}\right]$ of successfully grasping

4. We refer to Murphy (2012), Bishop (2006), and Hastie, Tibshirani, and Friedman (2001) for a thorough exposition on supervised learning methods.

(success denoted with S) a shirt, given some input image $\mathbf{x}$. Note that this example implies a heavy assumption; the availability of a dataset containing tuples of ⟨grasping pose, object configuration, probability of success⟩. In this situation, one could minimize the negative cross-entropy loss:

$$\mathscr{L} = -y^{(i)} \cdot \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \cdot \log \left(1 - \hat{y}^{(i)}\right),$$

with $\hat{y}^{(i)} = f(\mathbf{x}; \boldsymbol{\theta})$ being the predicted output of the model for observation i. The optimization problem then becomes to adjust the parameters $\boldsymbol{\theta}$ of the model f using the examples $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathbb{E}_{p(S,\mathbf{u},\mathbf{x})} \left[\mathscr{L}\left(S, Q_{\boldsymbol{\theta}}(\mathbf{u})\right)\right].$$

The dominant way for heavy parametrized functions such as neural networks to optimize this objective is to use gradient descent. The gradient expresses the direction of the steepest decrease of the loss function $\mathscr{L}$ with respect of the model parameters $\boldsymbol{\theta}$. By iteratively updating the parameters in the opposite direction of the gradient, in the case of a minimization objective, we gradually arrive at a local or global minimum:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathscr{L}(\boldsymbol{\theta}). \tag{2.1}$$

α determines how large steps we take towards the estimated direction of the closest local minimum. Adaptive methods such as Adam (Kingma and Ba 2014) allows taking variable step sizes per variable based on the historical directions of the gradient. Gradient-based optimization methods have proven to be a crucial for training highly parametrized functions such as deep neural networks, which are discussed in the following paragraphs.

2.3.2 Artificial Neural networks

ANNs are the workhorse of modern AI. ANNs are loosely inspired by the neural network in a biological brain and the mechanisms of learning in biological organisms. The human brain is build-up by interconnected processing units called neurons. The connection strength between these neurons changes in response to external stimuli. This way, neurons receive, process and send information through the body and brain of biological organisms. Although comparing artificial neural networks to their biological counterpart is criticized as a far-stretch from the inner workings of the human brain, insights and knowledge of the neuroscience field have been useful in designing neural network architectures. The most common computational model of neurons, visualized in Figure 2.7, simulates biological neurons as a node consisting of inputs, weights, bias, activation function and an output value. An ANN computes an output by propagating the computed values from the input neurons to the output neurons. The artificial neurons are connected through weights that scale the given input to the neuron. A single neuron performs a weighted sum of the inputs in order to arrive at the neuron's *activation*. Next, it transforms the activation value through an activation function before passing it to the successor neurons. The neuron's activation function is the source of nonlinearity in the network and enables the handling of non-linear relationships between inputs and outputs.

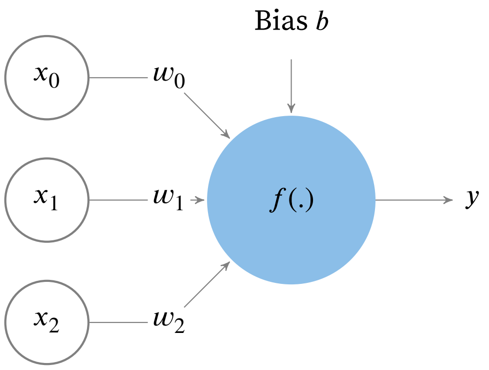


Figure 2.7 **Computational model of a single neuron.** x_0, x_1, x_2 represent input examples or signals from other units within the network. The bias term b represents an external input to the unit. The activation function is denoted by f and applied on the weighted input entering the unit.

More formally, the output y of an artificial neuron is computed by

$$y = f\left(b + \sum_{n=1}^{D} w_i x_i\right), \tag{2.2}$$

where f is the activation function, b is the bias, x_i is the ith input of the neuron which is weighted by weight w_i connecting the ith input. An example of an activation function f is the ReLU (Glorot, Bordes, and Bengio 2011) activation $f(x) = \max(0, x)$. Other common activation functions are sigmoid, hyperbolic tangents or variations of ReLUs.

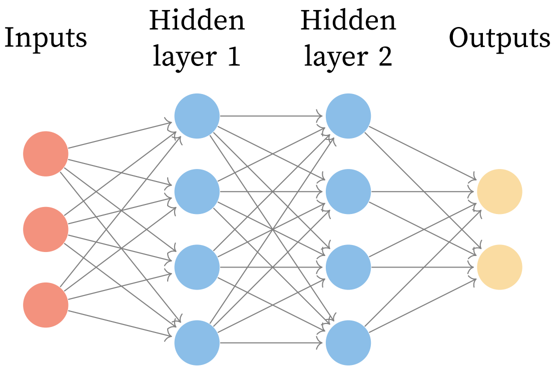


Figure 2.8 **A three-layered, feedforward, fully-connected neural network.** This particular network contains three inputs, two hidden layers and one output layer. All neurons of successive layers are interconnected but there are no connections within in a layer. The inputs are propagated from front to back.

In order to perform computations with neurons, we organize them in sequential layers. This gives rise to the name of artificial neural *networks* as they chain together many different functions in a directed acyclic graph. In fully-connected feedforward networks the input and output layer are separated by so-called *hidden layers*. The length of this chain is called the *depth* of the network. This architecture is visualized in Figure 2.8. The name of the layer often denotes the operation performed by the layer. For example, a softmax layer performs a *softmax* operation: normalize an input vector of real numbers to probability distribution proportional to the exponentials in the input numbers. Feedforward architectures propagate the inputs sequentially from layer to layer with neurons performing calculations as given in Equation (2.2) but in a vectorized way:

$$\mathbf{x}^{(k)} = f\left(\mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{x}^{(k-1)}\right). \tag{2.3}$$

In this Equation (2.3), the vector $\mathbf{x}^{(k)}$ contains the outputs of all neurons in layer k which is based on the input vector $\mathbf{x}^{(k-1)}$ from previous layer, multiplied with weight matrix $\mathbf{W}^{(k)}$. The computation occurring in the hidden layers solves a major problem in real-world problems where many applications require disentangling sources of variation by using high-level abstract features. Learned representations offer a solution for such problems because they often result in much better performance compared to hand-designed features. Stacking multiple hidden layers in a neural network allows learning representations based on raw data which solves this central problem of finding features at appropriate levels of abstraction. This paradigm of stacking layers of computational units is called deep learning. It takes a compositional learning approach: upstream representations are expressed in terms of other, simpler downstream representations. Early layers learn primitives which are combined later to form more complex features. In images for example, downstream neurons learn edges and corners which are used in upstream layers to learn to recognize for example the sleeve of a shirt.

Crucial in neural networks is that the representations are learned instead of crafted by hand. Learning in neural networks occurs by changing the strength of neurons' connections. The weight adjustment is a response to the network's error and has as goal to modify the computation to make the output maximize the given objective. Training of these network weights is done using gradient-based optimization as discussed in Section 2.3. Although alternative training methods exist such as evolutionary methods (Salimans et al. 2017), gradients provide the direction in which to change the weights in order to maximize the objective function. This is especially beneficial for highly parametrized functions such as neural networks. Differentiation of multi-layer neural networks, called backpropagation, was already figured out in (Rumelhart, Hinton, and Williams 1986) but was rediscovered by running the costly matrix-vector multiplication step in parallel on GPU (Oh and Jung 2004). Although improved computational hardware and data availability was a crucial enabler of the success of deep learning, many "tweaks" have proven as important to stabilize the backpropagation algorithm. Gradients assume an infinitesimal small step in each direction whereas the actual step we make has a finite length in order to make any real progress in optimization. The problem is that the gradients do change during the course of this step. In the case of multivariable optimization problems of considerable size, which is the case in deep neural networks containing millions of parameters, the optimization landscape is highly non-convex. This treacherous optimization landscape can change the gradients drastically leading to unstable training. This is why gradient-descent strategies such as momentum-based learning, using parameter-specific learning rates and weight initializing schemes are standard tricks in the deep learning practitioners toolkit. Gradients have also been known to disappear and diverge in deep neural networks because of repeated matrix multiplications when propagating the information forward and backward through the network. This is why ReLU activation functions are popular given that this piecewise linear activation has a derivative of value 1 in certain intervals and zero elsewhere. Another problem caused by the highly parametrized nature of deep neural networks is overfitting the data. In machine learning jargon, this means that a model can predict the training set well but performs poorly on hold-out samples. A popular way to deal with overfitting is to regularize the network weights. Regularization effectively reduces the network computational power by imposing a penalty on weights in the loss function:

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} P\left(f\left(\mathbf{x}^{(n)}, \boldsymbol{\theta}\right), y^{(n)}\right) + \Omega(\boldsymbol{\theta}). \tag{2.4}$$

In Equation (2.4), the function $P(.)$ is a chosen optimization metric that takes the output of the network $f\left(\mathbf{x}^{(n)}, \boldsymbol{\theta}\right)$ and the real label $y^{(n)}$ of sample n and is domain-dependent. The other term $\Omega(\boldsymbol{\theta})$ is the regularization term which balances L^1 and L^2 norm of the weights:

$$\Omega(\boldsymbol{\theta}) = \gamma \underbrace{\sum_k \sum_i \sum_j \left|\mathbf{W}_{ij}^{(k)}\right|}_{L^1 \text{ regularization}} + \lambda \underbrace{\sum_k \sum_i \sum_j \left(\mathbf{W}_{ij}^{(k)}\right)^2}_{L^2\text{-regularization}} . \tag{2.5}$$

The hyperparameters γ and λ in the regularization term of Equation (2.5) trade-off the amount of L^1 and L^2 regularization. The term $\mathbf{W}_{ij}^{(k)}$ refers to the weight in layer k connecting neuron i to neuron j. L^1 regularization achieves sparse weights while the L^2 norm leads to networks with smaller weights. Another popular method to improve generalization properties of deep neural networks is dropout (Srivastava et al. 2014), which zeroes out different random neurons at training time, ensembling and using data augmentation.

The feedforward models described above connect the neurons between layers in a fully-connected manner: every neuron from a layer is connected to every neuron from the preceding and succeeding layer with a unique weight. This dense connectivity leads to an explosion in the number of trainable parameters. However, when the input data contains topological structure, like the ordering of image pixels in a grid, constraining the connectivity pattern between layers is a useful method to reduce the number of parameters and exploit correlation. The most common way to implement this is by replacing the matrix-vector product $\mathbf{W}^{(k)}\mathbf{x}^{(k-1)}$ of Equation (2.3) with a sum of convolutions. This operation is equivalent to sliding a low-dimensional filter or kernel over the input image while performing a dot product. This property leads to sparse connectivity and parameter sharing. Consequently, by connecting a local spatial region with a shared set of parameters to the full spatial resolution of the image, one imitates the cortical neurons in the visual cortex, which respond only to stimuli within a receptive field (Hubel and Wiesel 1959). It has been argued that these properties explain the success of using trainable convolutions for computer vision (Goodfellow, Bengio, and Courville 2016). Usually, the convolution operation is followed by a downsampling operation that provides a summary statistic of the nearby outputs. This is most frequently implemented with an aggregation function, for example max pooling that takes the maximum value of a rectangular neighborhood. The process of embedding convolution operations, optionally followed by pooling operations, is known as a CNN. After demonstrating the effectiveness of CNNs for large-scale image classification (Krizhevsky, Sutskever, and Hinton 2012), using deep hierarchical neural architectures have been omnipresent in computer vision and natural language processing. In robotics, convolutional layers can provide a perception module while the motor control module is implemented as fully-connected layers acting on the output of the filter banks. However, in the case of motor control, the pooling operation is ofted removed from the architecture as translational invariance is not a desired property. Invariance to the position of a detected object would not enable a robot to detect where the object is located in the image. For more details about the history and working of deep learning methods and architectures, we refer the reader to the textbook of Goodfellow, Bengio, and Courville (2016).

Supervised learning is an important paradigm for robotic learning because labelled data provides a clear learning signal. This is important because time spent on robots is expensive. On the contrary, the learning signal in reinforcement learning (Section 2.5) often does not optimize for direct task performance and might lead to expensive learning times on physical robot platforms. The following paragraphs discuss relevant work in applying machine learning methods for solving robotic manipulation tasks.

2.3.3 Supervised learning and neural networks in robotic manipulation research

Traditionally, supervised learning methods are leveraged in the **perception module** of a robotic manipulation pipeline. For rigid body manipulation, the popularity of data-driven grasp synthesis approach took off with the work of Saxena, Driemeyer, and Ng (2008). In their work, a logistic regression classifier is trained on synthetic data using manually engineered features. This way, they demonstrate that a robot is able to unload a dishwasher. In the domain of deformable object manipulation, Ramisa et al. (2012) searches for quality grasping points of crumbled cloth in order to maximize the unfolding upon lifting. They do this by first labelling a dataset of shirts with bounding boxes containing the appropriate grasping points. Next, they train a logistic regression model to obtain the probability of the desired grasping point in a given bounding box using a bag of features from the input image. By employing the logistic classifier in a sliding window over the image, they pass image patches containing local peaks to an SVM to obtain more accurate grasping candidates. The candidate patch is converted to 3D space by working in a calibrated environment, and motion planning is executed using inverse kinematics. Similarly, P. C. Wang et al. (2011) folds socks by having a perception system that uses manually engineered features for training an SVM with Gaussian kernel to determine the type of sock in front of the robot. SVMs have also been used to identify garment category and pose (Li, Chen, and Allen 2014; Yinxiao Li et al. 2014). The deformable nature of cloth leads to self-occlusions making the garment type and pose classification ambiguous. The presence of this hidden state is explicitly modelled using a HMM in (Cusumano-Towner et al. 2011). Learning methods are also used to find regions of interest on the cloth. For example, Doumanoglou et al. (2016) use random forests to learn garment-specific grasping points. In (Maitin-Shepard et al. 2010), RANSAC (Fischler and Bolles 1981) is used to find the corners of the cloth. These corners are good candidate grasping points for unfolding and identifying the type of cloth. Finally, nearest neighbours have been used to identify wrinkled regions in a washcloth in order to flatten it (B. Willimon, S. Birchfield, and I. Walker 2011).

The earliest work using neural networks for deformable object manipulation is (Howard and Bekey 2000). In their work, they train a small feedforward neural network that learns the required minimum grasping force for lifting a deformable object. They collect the data by iteratively using more lifting force on objects with certain masses, deformability and damping. With the breakthrough in deep learning in 2012 by Krizhevsky, Sutskever, and Hinton (2012), deep neural networks have found their way into robotic manipulation, starting in the rigid body manipulation domain. A successful approach to training deep neural networks in a supervised setting for robotic manipulation is to use CNNs as grasp success predictor. Levine et al. (2016) train a CNN on a large dataset of 800.000 grasping attempts to learn to predict the grasp success probability of a grasping pose, given an input image. To sample candidate grasping points, they employ CEM (Rubinstein and Kroese 2004). Dex-Net (Mahler et al. 2017a) also trains a CNN to predict the quality of a grasping candidate. This network is trained using a simulated dataset where objects are put into randomized poses on a plane. They use simulation to evaluate different grasping wrenches using analytic grasp metrics. Their model shows impressive generalizability to the real world, on different models not seen during training. The Dex-Net framework has been extended to work with suction grippers (Mahler et al. 2017b), use dual-armed robots (Mahler et al. 2019) and generate grasping candidates in the network (Satish, Mahler, and Goldberg 2019).

Another strategy to train controllers using supervised learning is **behavioural cloning**. Behavioural cloning is a type of imitation learning (Argall et al. 2009) in which task demonstrations are used for learning task execution. In behavioural cloning, a sequence of states and actions, as executed by a demonstrator, is recorded as dataset for a supervised learning algorithm. The goal then becomes for the model to predict the action a demonstrator would choose, given a certain state. An in-depth view of the field of learning from demonstration is given in (Argall et al. 2009). In the case of robotic laundry, Jia et al. (2019) learns single robotic laundry tasks for flattening, folding and twisting cloth by imitating human examples. Notably in their work is representing the robotic controller using random forests (Breiman 2001) instead of neural networks. The rationale is given by the non-parametric nature of random forests to dynamically change the number of leaf nodes based on the given imitation data and new cloth configurations. Example demonstrations are also used in deformable object manipulation to tie knots. Schulman et al. (2016), for example, uses example demonstrations for non-rigid warping (Chui and Rangarajan 2003) based on point-cloud registration of the scene to tie knots with a robotic manipulator. The general idea is to warp the demonstrated trajectory to match the current setting, which may vary in initial conditions and knot geometry. Closely related is the work in (Morita et al. 2003) where examples and solutions strategies from knot theory is embedded to perform motor control. Neural network models have also been a strong candidate for robot learning from demonstration (Ravichandar et al. 2020). Given the cost associated with real robot rollouts, it is beneficial to lower the dataset requirements by having access to the motor control outputs or to decrease the input dimensionality by, for example, avoiding learning from pixels. This explains the popularity for teleoperated (T. Zhang et al. 2018; Duan et al. 2017) and kinesthetic teaching (Finn et al. 2017) approaches. However, it is difficult to teleoperate a robot or physically manipulate a robotic arm to fold clothing items. This is because the speed and forces associated while executing the task is relevant for achieving proper folds. One method to compensate for this difficulty is to solve the imitation learning task in simulation and trying to transfer it to the real world. This is explored in (Seita et al. 2020) that generates example demonstrations in simulation and uses behavioural cloning to train a motor policy network to flatten a towel. To fine-tune this policy outside the seen dataset distribution, they employ dataset aggregation (DAgger) in which an oracle policy is used to label the unseen states during training. An alternative is given in (Sundaresan et al. 2020) that uses simulated data to learn visual object descriptors indicating the segments of a rope. Such embeddings implicitly encode geometric structure which can then be used for knot-tying using example demonstrations. Although imitation learning is a viable alternative for learning to manipulate objects, a general problem plaguing imitation learning methods is generalizing to unseen scenarios. In the case of trajectory execution, errors can accumulate which drastically increases the probability of task failures. This is why existing methods apply data augmentation, include teacher advice or use reinforcement learning (Section 2.5).

Another approach to leverage the expressiveness of deep neural networks in a supervised setting is to train a **dynamics model**. This model, often called world model, is obtained by training on $\langle$state, action, next state$\rangle$ tuples. The problem that world models try to solve is finding the optimal sequence of actions that brings the given input state to the desired output state by querying the world model. Or in other words, the robot knows *how* to act while the user can tell the robot *what* it should do. To manipulate a rope into the desired shape, for example an S-shape, A. Nair et al. (2017) let the robot make arbitrary manipulations on the rope while recording the state transitions. This data is then used for training the inverse dynamics model. Planning can then be done using the world model by giving transitionary keyframes that define subgoals to the robot. A similar world model is trained in (Ebert et al. 2018) where the model learns to predict future pixels given the planned actions. The data collection is done in a self-supervised way in which the robot does motor babbling. The trained world model then enables model predictive control in which they show good performance for folding cloth and towels. Whereas the training data for the world model in (A. Nair et al. 2017) is given by self-supervision, in (Yang et al. 2016) the data is given by teleoperating a humanoid robot with a virtual reality headset. This training data is then passed into an autoencoder which produces a time series in latent space. A second stage neural network then learns cloth dynamics by sliding a window over the encoded time series. A further attempt to embed the world model into the control module can be done by sandwiching a fully-connected network between the encoder and decoder (Tanaka, Arnold, and Yamazaki 2018). Instead of embedding a controller module, other work embed physics priors into the network architecture. By assuming the deformable objects are build-up of small, connected particles, it is possible to represent their connectivity and interactions in a graph neural network. This allows to efficiently learn deformable object dynamics to solve for downstream control tasks such as poking deformable objects (Mrowca et al. 2018) and merging liquids (Yunzhu Li et al. 2018).

Main finding

Label-based learning for robotic manipulation tasks is efficient but not always effective or possible.

To conclude, framing robotic manipulation of rigid and deformable objects as a supervised learning problem has been successful for estimating the state of deformable objects or manipulating cloth by means of behavioural cloning. However, training with hand-labelled or generated data has two main issues (Pinto and Gupta 2016). The first issue is the human bias towards preferring grasps poses that are similar to the way a person would grasp an object. This discourages the exploration of unconventional grasp configurations. The second issue is the cost to exhaustively evaluate all possible grasps because an object can be grasped in multiple ways. Therefore, learning on robots requires a method that can work without human supervision.

2.4 Learning robotic manipulation tasks without user-provided labels

Annotating datasets is an expensive effort that requires manual labour in order to ensure high-quality labels. For example, labelling all wrinkles in a cloth of a human folding clothing in a video of 60 s recorded at 60 fps, requires annotating 3600 frames. Within machine learning, there exists a body of research that avoids this costly annotation effort. We discuss two approaches of learning without manually provided labels, using unsupervised and self-supervised learning methods, next.

2.4.1 Unsupervised learning

Whereas supervised learning methods search for a mapping between inputs and labels of observations, unsupervised learning uncovers structure in the input data without the use of any labels.

Unsupervised learning

Unsupervised learning extracts regularities from unlabelled datasets in order to reduce the description of the data to their most characteristic elements.

The first class of unsupervised learning applications is *clustering*: segmenting data into similar groups based on a similarity metric that calculates the distance between observations. In the case of robotic folding, clustering is a popular method for segmenting an input image in order to separate different clothing articles in the scene (Doumanoglou et al. 2016; Maitin-Shepard et al. 2010; Jia et al. 2018). The second class of unsupervised learning is dimensionality reduction methods, which reduces the dimensionality of the features of a dataset. Feature extraction, in particular, looks at projecting the original input features to a new space of smaller size while preserving as much of the significant structure of the input space as possible. Reducing the input dimensionality is important for machine learning algorithms because their complexity not only increases with the size of the input dataset but also with the number of features. The practice of feature extraction is present in the deformable object manipulation domain, given that high-dimensional camera streams are often used as a sensor for estimating the state of cloth. Yinxiao Li et al. (2016) for example, considers the problem of cloth flattening for robotic ironing. To reduce the dimensionality of the image of a discontinuity scan of a cloth, they employ SIFT descriptors (Lowe 1999) as input for a support vector machine classifier to assign the probability of a discontinuity to be a permanent wrinkle. Other work (Jia et al. 2018) observes that deformations on cloth can be detected as shadows and shape variability. To exploit this visual property, they apply a set of Gabor filters on the input image and accumulate the filtered images into a histogram as a high-level state representation of the cloth. A noticeably early work of using neural networks is the work of Foresti and Pellegrino (2004) in which the problem of grasping fur tails from a conveyor belt is considered. To segment the different furs present in the image, they train self-organizing maps (Kohonen 1982); a neural network in which neurons compete to be activated by the input signal. This results in disconnected regions of interest that are joined using skeletonization. Finally, a heuristic is used to determine and grasp the largest fur. Autoencoders, i.e. neural networks trained with reconstruction tasks, are another useful feature extractor. Yang et al. (2016) for example, use the latent space of a deep convolutional autoencoder as input for a deep fully-connected neural network that acts as a dynamics model of the cloth. Depth sensor streams is another high-dimensional input modality for cloth state estimation, which can be reduced to a lower dimension. Ramisa et al. (2013) for example, transform depth images to SIFT-like descriptors that describe a patch based on the distance between the normals in the patch and a reference set of normal directions.

Together with data preprocessing, dimensionality reduction methods also play an important role in data visualization. The analysis of the underlying structure of a high-dimensional dataset can be facilitated when the data is represented in fewer dimensions. In the context of deep learning, non-linear dimension reduction methods are being used to study the semantics of what the hidden units are encoding. These methods, in particular, look at reducing the input space to the 2D plane in order to enable the plotting of the underlying patterns in the data. Amongst non-linear dimension reduction methods, t-SNE and UMAP are often chosen techniques for visualization purposes. Both methods construct a graph and optimize a subsequent low-dimensional embedding that preserves the structure of that graph. At the core, these methods employ loss functions that make similar points attract each other and push dissimilar points away. From a high-level perspective, t-SNE defines distances between samples as conditional probabilities and optimizes a low-dimensional embedding in which the relative distances between samples match those of the original high-dimensional distances. This optimization is defined as minimizing a Kullback-Leibler divergence between the pairwise distances ins the low-dimensional embedding and the high-dimensional input space. t-SNE is, for example, being used in TCNs of Sermanet et al. (2018) to show that the learned embeddings semantically encode the same pose of robots and humans close in embedding space. For example, both images of a robot and a human crouching and extending the same arm are encoded proximate in embedding space. Similarly, in Atari DQN (Mnih et al. 2015) the last hidden layer is represented with t-SNE embeddings to discover that visually dissimilar states are close in embedding space due to having the same expected reward. UMAP is another state-of-the-art dimensionality reduction method that has been successfully employed in biology (Cao et al. 2019), machine learning (Carter et al. 2019) and social sciences

(Diaz-Papkovich et al. 2019) to discover underlying structures in high-dimensional datasets. UMAP constructs a weighted graph of nearest neighbours with the weights representing the probability that two points are connected. Then, UMAP optimizes a low-dimensional representation of this graph that is structural as similar as possible. This is done by minimizing the cross entropy in order to measure the distance between the high-dimensional and low-dimensional graph. UMAP has been argued to better preserve global structure compared to t-SNE (Becht et al. 2019) resulting in semantically more meaningful clusters.

2.4.2 Self-supervised learning

A large class of successful ML methods rely on some form of supervision. In robotic folding, this supervision can be labelling the type of clothing item in front of the robot, specifying which action a human would take or which torques to exert on the motors. Collecting this data is expensive, sometimes difficult and prone to bias and errors (Mehrabi et al. 2021). Unsupervised learning is on the other end of the *supervision spectrum*, but it is hard to know which signal it will pick up for learning. Providing supervision while building up a dataset, without manually labelling every sample, is known as *self-supervised learning*. Therefore, by having a data generation process that creates pseudolabels, one can use well-established supervised learning algorithms. The labelling process allows us to inject prior knowledge about the task by inventing mock tasks, also known as *pretext tasks*. The goal of self-supervised learning is not to solve this invented task but rather to learn meaningful representations that can be used for downstream tasks. In cloth folding, for example, we can present the network with a shirt, rotate it and ask the model to predict how many degrees the shirt was turned. Although this example task is only deployable in a narrow set of tasks, we do not care about the accuracy. Instead, we want the network to learn a meaningful latent space.

There are two broad categories of self-supervised learning. The first category, generative methods, aims to reconstruct the input signal while learning a latent space. These methods are popularized by generative adversarial networks (Goodfellow et al. 2014). The second category, discriminative methods, reframes the self-supervised problem as a classification task. In particular, *contrastive* self-supervised learning looks at *contrasting* positive and negative examples. More formally, let $h(\mathbf{x}) \in \mathbb{R}^d$ be the encoding of an input sample $\mathbf{x}$, called the anchor. We define $\mathbf{x}_p$ as a positive sample and $\mathbf{x}_n$ as a negative sample. The goal then becomes to encode the anchor and positive close in embedding space compared to the anchor and the negative:

$$\operatorname{dist}\left(h(\mathbf{x}), h(\mathbf{x}_p)\right) \leq \operatorname{dist}\left(h(\mathbf{x}), h(\mathbf{x}_n)\right).$$

The distance function dist measures how similar samples are. For example, in the case of robotic folding, we would want the embedding to push frames of folded shirts closer together than completely crumbled shirts. The hypothesis is that by forcing similar items to be close in embedding space, the network has to learn relevant features and build a semantically meaningful embedding this way. This is done by enforcing invariances in the embedding. For example, if the network is presented with an anchor image, a rotated anchor image and a random different image, the network learns to become rotational invariant. Other popular pretext tasks in literature to learn different types of invariances are learning to colourize images (Zhang, Isola, and Efros 2016), reconstructing the original input (Pathak et al. 2016) and predicting the relative position of two random patches (Doersch, Gupta, and Efros 2015).

Self-supervised learning

Self-supervised learning aims to learn semantically meaningful embeddings by solving mock tasks that force the model to attend to task-relevant features while learning useful invariances.

Using contrastive objectives to learn good representations have been popularized in the NLP domain. In (Mikolov et al. 2013), contrastive training was done by using co-occurring words as semantically similar for learning word embeddings. Extensions to images, video and speech data were popularized with contrastive predictive coding (van den Oord, Li, and Vinyals 2018).

This method maximizes the mutual information between the predicted encodings and their corresponding positive samples. Similarly, SimCLR (Chen et al. 2020) maximizes agreement between an image and its applied data augmentations by using a cosine similarity loss.

In this work, we focus on extracting self-supervised signals from video task demonstrations. The inherent structure in videos is the temporal dimension, which can be used as a supervisory signal to provide contrasting examples. The goal then becomes to recover the temporal coherence of a video. One of the first works (Misra, Zitnick, and Hebert 2016) leveraging time as contrastive signal inputs a sequence of frames and classifies whether the frames are in the correct order. Later work (H.-Y. Lee et al. 2017; Fernando et al. 2017) also frames self-supervised learning as a classification task in which the correct temporal order has to be determined. Follow-up work has looked at using self-supervised embeddings as a reward or progression signal for learning agents. In Singh et al. 2019, they construct a reward function based on an image classifier trained on successful goal states reached by teleoperating the robot towards the end state. In Hartikainen et al. 2019, time is used as a learned distance function for assigning environment rewards. However, their approach requires human intervention in order to select the desired goal states. S. Nair et al. 2020 also uses time as a supervisory signal in videos of expert demonstrations to learn an optimal trajectory of states. However, they assume the possibility of visually removing the end-effector from the scene which is not possible for all tasks. For example, it is not possible to drop a shirt in mid-air while being folded. Other work A. V. Nair et al. 2018 looks at expressing the reward function as the distance in latent space between the current state and the goal state. However, this is not possible when there is a trajectory in latent space that has to be followed in order to execute the task. This problem is enlarged when the start state is very similar to the end state. In this scenario, the agent would not be incentivized to leave the start state because it already receives high rewards due to being close to the end state. Alternatively, it is possible to add a contrastive loss as an auxiliary objective in RL as done in CURL Laskin, Srinivas, and Abbeel 2020. The authors show that their method outperforms other learning methods on the DM Control Suite, i.e. a set of continuous control tasks in simulation. An important self-supervised architecture we use in this work are TCNs (Sermanet et al. 2018). The central idea in this approach is to leverage cross-modal inputs, such as different camera viewpoints, in order to find differences in frames that cannot be attributed to a changing viewpoint. We discuss this in detail in Chapter 6.

In the domain of deformable object manipulation, self-supervised methods have been used primarily to do the state estimation of objects. A latent space describing the state of a rope is learned using self-supervised learning on simulated images in (W. Yan et al. 2020). This model is then used as forward dynamics model in which an action trajectory is sampled that minimizes the distance towards the given goal state. The actions are chosen such that only points on the rope are considered. A similar approach for bringing fabric into a desired state is explored in (Hoque et al. 2020). They learn an image prediction model in simulation, which can be used to solve arbitrary goals at test time via model predictive control. They apply domain randomization to transfer fabric smoothing policies to a real-world surgical robot. However, their approach fails to transfer folding tasks successfully due to the sim-to-real gap. An alternative sim2real solution is explored in (M. Yan et al. 2020) that also train an autoencoder to predict the dynamics of deformable linear objects. The encoder uses a transformer architecture (Vaswani et al. 2017) that iteratively refines the estimation of the image space coordinates of points on the rope. The network is then fine-tuned on real images using a self-supervised objective. This self-supervised objective encodes a colour contrast cue: the target deformable linear object has a different colour from the background. This prior can be modelled with a Gaussian mixture model that segments the input image. By rendering the output rope state from the network to image space, a differentiable loss can be defined on the segmented input image to refine the network. This method leverages simulated data for pretraining of the network, approximately 5000 real images for fine-tuning and is able to accurately manipulate a rope into a given target state.

2.5 Learning robotic manipulation tasks from interaction

2.5.1 Reinforcement learning

In supervised learning, a clear learning signal allows training models such that their output matches the labels given in the training set. This requires obtaining a dataset that labels every observation with the correct response. For example, in the case of a robot folding a shirt, supervised learning requires providing supervision on how to move every joint for all arm and shirt configurations in the training set. This sequential decision-making process might alternatively better be solved by providing an indication on how good the model is behaving. For example, we might give a robot a positive reward when it has folded a shirt without wrinkles and penalize it when it throws the shirt of the table. This approach of giving agents rewards in an environment is formalized as RL and is an eminent approach for learning control policies with minimum user intervention. RL has been successfully applied in domains ranging from game playing (Mnih et al. 2015), helicopter flight (Ng et al. 2003), autonomous driving (Sallab et al. 2017), locomotion (Tan et al. 2018) and robotic manipulation (Levine et al. 2016).

Reinforcement Learning

Reinforcement learning is interactively observing and influencing the environment while receiving rewards in order to learn what to do in order to solve a task.

Sequential decision making is formalized as a MDP: given a sequence of states, determine the action that will maximize the expected discounted future reward. This statement implies that an agent can observe its environment, exert influence on its environment through actions and has a notion of what constitutes good behaviour. More formally, an MDP is a tuple (S, A, P, γ, R) where:

1. S is a set of states representing the environment. For example, the set of joint values of a robot arm and the position of

the sleeve of a shirt.

2. A is a set of actions. These actions are performed by the agent and influence the environment. For example, moving the end-effector of the robotic arm up or downwards.

3. P are the state transition probabilities. This is a distribution over states and actions indicating the probability to arrive at a new state: $P\left(s_{t+1} \mid s_t, a_t\right)$.

4. $\gamma \in [0, 1[$ is the discount factor and used to discount future rewards to the present.

5. $R = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$ is the reward function, based on the action taken and the resulting state the environment transitions to.

Decision making in an MDP goes as follows: the environment is initialized in a certain state s_t. The agent chooses an action a_t to execute in the MDP. This way, the state of the MDP transitions to a successor state s_{t+1} governed by P. Based on this new state, the agent receives a reward r_{t+1}[5] and chooses a new action a_{t+1}. This process is visualized in Figure 2.9 and repeats until the environment signals a terminal state, for example, a robot manipulator successfully folds a shirt. MDPs with terminal states are called *episodic* MDPs. Furthermore, a distinction is made between *states* and *observations*. A state is assumed to contain all relevant information to make an optimal decision. In technical jargon, this is called the *Markov property*: the future depends only on the current state and action, but not on the past. However, it is hard to capture all task-relevant features with sensors. For example, a single camera image of a crumbled shirt does not satisfy the Markov property because the occlusions do not allow to make informed decisions about the occluded parts. In such cases, the states are called *observations* O and the formalism a POMDP.

The eventual factor driving the decision of the agent is the reward function. The goal of an RL agent is to maximize the sum of

5. Both notation r_t and r_{t+1} are used in literature. We use r_{t+1} to denote that s_{t+1} and r_{t+1} are jointly determined.

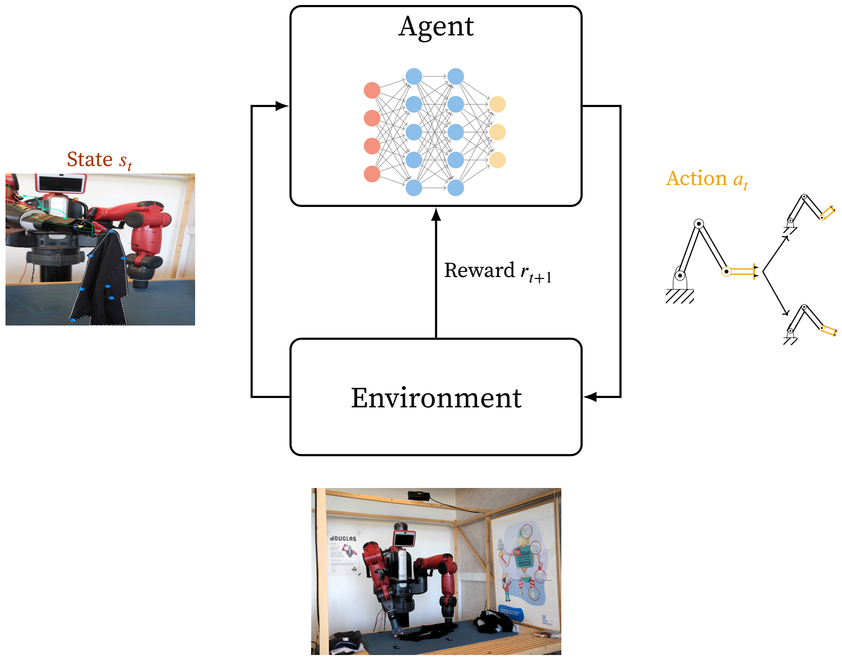


Figure 2.9 Canonical flow of the RL loop in the context of robotics.

expected discounted rewards G_t:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T \\ &= \sum_{k=t}^{T} \gamma^{k-t} R_{k+1}. \end{aligned} \tag{2.6}$$

To cope with the difference between the immediate reward $R_t(s_t, a_t)$ of a state-action pair and the long-term value of taking action a_t in state s_t, we introduce the notion of *value functions*. A value function denotes how good it is to be in a certain state. More formally, the value function $v_\pi(s)$ expresses the expected, cumulative, discounted, future reward of a state:

$$v_\pi(s) = \mathbb{E}\left[G_t \mid S_t = s\right], \tag{2.7}$$

or of a state-action pair:

$$q_\pi(s, a) = \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right]. \tag{2.8}$$

The function $q_\pi(s,a)$ is called the Q-function. This notation introduces the policy π in the subscripts $v_\pi(s)$ and $q_\pi(s,a)$. This is due to agents' rewards depending on which actions they will take in the future. A policy is a distribution over $a \in \mathscr{A}(s)$ for each state $s \in \mathscr{S}$: it maps the probability of selecting an action given a certain state. Hence a Q-value denotes the total reward the agent receives in state s_t for taking action a_t and then following policy $\pi(s,a)$ in expectation. The goal of the agent then becomes to find the policy that maximizes the value functions:

$$v_*(s) = \max_\pi v_\pi(s) = \max_a q_{\pi^*}(s,a). \tag{2.9}$$

Writing out the expectation of Equation (2.7) gives rise to the Bellman equation:

$$v_*(s) = \max_a \sum_{s'} p\left(s' \mid s,a\right)\left[r + \gamma v_*\left(s'\right)\right], \tag{2.10}$$

with s' being the successor state of state s after taking action a. In the case of Q-values the Bellman equation becomes:

$$q_*(s,a) = \sum_{s'} p\left(s' \mid s,a\right)\left[r + \gamma \max_{a'} q_*\left(s',a'\right)\right]. \tag{2.11}$$

The Bellman equation provides recursive decomposition and value functions that allow to reuse sub-solutions. Hence, the Bellman equation can be solved using dynamic programming. However, dynamic programming algorithms, such as value iteration and policy iteration (Sutton and Barto 2018), requires knowing the transition and reward models of the MDP. Modern RL algorithms are often divided on a spectrum ranging from exclusively finding the value of states to finding how to behave optimally without inferring any value function. This distinction is discussed next.

Value-based methods The first category of RL algorithms are value-based methods, also called critics, which try to discover the value function by interacting with the environment. This principle is known as TD learning (Samuel 1959) and is argued to be the most unique contribution to RL (Sutton and Barto 2018).

TD learning combines the ideas of dynamic programming (problem decomposition and reusing subsolutions) and Monte Carlo methods (averaging complete returns by sampling the probability distribution). The central idea is to learn the value function directly from experience with bootstrapping, in a model-free and online manner. A popular instantiation is Q-learning (Watkins and Dayan 1992) which iteratively updates the Q-function as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[\underbrace{r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)}_{\text{TD error}} \right] \tag{2.12}$$

Representing the value function $Q(s,a)$ has originally been done with tables (Watkins and Dayan 1992) but do not scale to real problem state and action spaces. The memory and time required to fill a table representing all possible states of, for example, a camera image are infeasibly high. This is where reinforcement learning borrows the idea of supervised learning, or more generally function approximation, from the machine learning domain. By using machine learning models as Q-function representations, an agent can use seen data from interaction with the environment to learn to generalize to states never seen before. A broad handling and discussion of both tabular and approximate RL methods is discussed in the textbook of Sutton and Barto (2018). Neural networks, in particular, have been early candidates as function approximators for value functions, often called Q-networks. Gradient-based optimization of the Q-network is then possible by backpropagating the TD error indicated in Equation (2.12). Neural networks are notoriously difficult to train in a TD setting because updating weights has unpredictable changes at other places in the state-action space. Mitigating this non-stationary property of the data has been central in making value-based RL methods work on real-life tasks. In the context of control, Riedmiller (2005) developed NFQ in which a three-layered feedforward neural network is trained to predict the state-action values of control tasks such as cartpole regulation. NFQ is characterized by using a replay buffer (Lin 1992): storing the experience tuples (s_t, a_t, r_t, s_{t+1}) in memory for training purpose. NFQ employs this replay buffer in a batch RL manner: the neural network is trained from scratch on the whole replay buffer each time the training loop is called. Scaling this system up to working with image-based inputs using CNNs is called DQN (Mnih et al. 2015). DQN learns online by sampling the replay buffer and computes future Q-values ($r + \gamma \max_{a'} Q\left(s', a'\right)$) in Equation (2.12)) with older versions of the Q-network, called the target network. A major problem with value-based methods is that they do not optimize the task objective directly. Instead, value-based agents search for the correct value of state-action pairs instead of searching how to behave optimally. This problem is solved by the next category of RL agents, policy-based methods, which is discussed in the next subsection.

Policy-based methods The second category of RL algorithms is labelled policy-based methods, also called actor and actor-critic methods[6], which optimize a parametrized policy directly. Assuming a differentiable policy $\pi_{\boldsymbol{\theta}}(a \mid s)$ parametrized by $\boldsymbol{\theta}$, the goal of policy search becomes to find the optimal policy parameters $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{\tau} \sim P_{\boldsymbol{\theta}}(\boldsymbol{\tau})} \left[\sum_{t}^{T} r(s_t, a_t) \right]. \tag{2.13}$$

In this notation, $\boldsymbol{\tau}$ denotes a trajectory of state-action pairs $(s_0, a_0, s_1, a_1, \cdots)$. Parametrization of the policy offers a natural way to deal with applications in continuous action spaces, compared to value-based methods that require evaluating all actions. Policy-based methods also optimize for task performance directly. However, policy-based methods are less sample efficient compared to value-based methods because new data needs to be sampled each time the policy changes. Common policy search methods are A3C (Mnih et al. 2016), PPO (Schulman et al. 2017) and TRPO (Schulman et al. 2015).

Value-based methods have borrowed ideas from policy-based methods to extend Q-learning to a continuous action space.

6. Historically, a distinction was made between actor methods and actor-critic methods depending on whether a critic with parametrized value function approximation was used. However, modern policy-based methods all use some form of value function approximation rendering this distinction obsolete.

Notably, DDPG (Lillicrap et al. 2015) and SAC (Haarnoja et al. 2018) which learns a parametrized policy such that the max operation in DQN of Equation (2.12) is differentiable with respect to the action argument: $\max_{a'} Q_\phi(s, a') \approx Q_\phi(s, \mu_\theta(s))$, with Q_ϕ being the value function neural network parametrized by ϕ and μ_θ the policy neural network parametrized by θ. DDPG have been succesfully used to train a seven degrees of freedom arm for reaching and door opening tasks (Gu et al. 2017).

Comparison to supervised learning. The sequential decision making of RL introduces some challenges not present in supervised learning. In RL, the consequences of actions are often delayed, making it hard to assign actions to outcomes. For example, if a robot fails to fold a piece of cloth after 100 steps, it is hard to know whether this was due to the end-effector losing grip of the cloth at step 99 or the fast motor accelerations at step 42. This problem is known as the credit assignment problem. Additionally, the sequential nature of RL makes it difficult to reuse data, making RL more data-intensive. This is due to many RL algorithms having an *on-policy* nature: each time the policy that we want to optimize changes, we need to collect new data of it. Another aspect of decision making is trading-off exploration and exploitation: if the robot knows that folding a shirt via the sleeves leads to success, why would it try other methods such as lifting the shirt in the air and risk failing? This exploration-exploitation dilemma has been coped with by introducing randomness in policies or curiosity (Pathak et al. 2017) but remains an unsolved problem (Sutton and Barto 2018).

RL in the context of optimal control. RL and classical optimal control both address the problem of finding an optimal policy that optimizes an objective function in a system described by states, actions and a model governing the state transitions. However, optimal control assumes perfect knowledge of the state transitions while RL operates directly on measured data (i.e. rewards from environment interaction). RL fills the gap in optimal control for tasks that are analytically intractable and can be viewed as adaptive optimal control (Sutton, Barto, and Williams 1992).

RL in the context of robotics. Much of the work for learning manipulation skills for deformable objects has been inspired by work done in the rigid object's domain. These methods, in turn, often rely on work done on learning to play games and solve toy tasks in simulation environments. First attempts at transferring vision-based DQN to visual servoing were attempted in (F. Zhang et al. 2015) but failed. Their end-to-end approach to frame the inherent continuous action spaces of robotic manipulation to a discrete algorithm such as DQN is to discretize the joint motor outputs into 9 motor actions bins. To further reduce the search space, they reduce the number of joints of the robot from 7 DOF to 4 DOF. They train a DQN controller in simulation while adding noise and task variations. However, transfer to the real robot failed due to the low simulation fidelity and inappropriate reward functions. Later work (James and Johns 2016) managed to transfer virtually trained DQN agents on pixels by using high-fidelity simulation and careful reward tuning. With the introduction of continuous action spaces for value-based agents, more specifically DDPG (Lillicrap et al. 2015), DQN variants started being successfully applied in the robotic manipulation domain (Gu et al. 2017). This success was mainly driven by directly representing the continuous action space of robot actuators, reusing past experiences for training and working in parallel.

Deep RL for deformable object manipulation has arguably first been used in (Matas, James, and Davison 2018). By training a DDPG agent in simulation, they demonstrate limited transferability to the real world using domain randomization to fold a towel. They find that a lack of simulation tools for cloth is the main factor limiting their results. The same approach is improved in (Jangir, Alenyà, and Torras 2020) by warm-starting the learning with example demonstrations. An additional speed-up can be obtained by filling the replay buffer with example demonstrations (Tsurumine et al. 2019). Wu et al. (2020) examines how to link two policies, i.e. picking and placing cloth which are normally independently trained. They opt for structurally encoding the relationship between picking and placing of cloth. This is done by having the second learned policy, i.e. the place policy, receive the last output of the picking policy. This way, the network can optimize the picking point that gives the most value during placing. Batch RL methods, a variant of RL in which the agent is trained once on a dataset without interacting with the environment, is explored in (R. Lee et al. 2020). Instead of generating an interaction dataset with random motor actions, they use a motor control heuristic. Such heuristics allow collecting data that contains more meaningful interactions compared to motor babbling. They train a fully convolutional DQN agent offline, resulting in a Q-value heatmap per possible action. This approach is successful in achieving simple folds in rectangular cloth, with one hour of real-world data. Another DQN architecture is explored in (Seita et al. 2021). They employ Transporter networks, an architecture that predicts the spatial displacement of a local area, to learn to fill deformable bags with objects in simulation.

Transitioning from solving problems with supervised learning to using RL carries problems concerning data efficiency, generalizability, delayed rewards and exploration-exploitation trade-off. These problems are enlarged in the robotics domain. Experience on a real physical system is expensive to obtain, so reusing past data and trading-off exploration and exploitation becomes even more important. Additionally, while RL is already faced with reproducibility issues (Henderson et al. 2018), reproduction and repetition on real robots are non-trivial due to robot wear, noisy state observation and stochastic action execution. This real-world reproducibility issue also makes it difficult to establish a common ground for benchmarking RL algorithms on physical robots. A category of RL algorithms improves the sample efficiency of RL by learning the system dynamics from interaction. However, small errors under this model-based RL approach accumulate, making it hard to transfer the results to the real world. Similarly, latency issues also plague effective learning: delays between the real state and state observation and between the action choice and action execution requires some form of memory in order for the state to be Markovian. A problem often neglected in the general RL community is safe exploration: a cloth folding robot, for example, might tear a shirt to pieces. Finally, although RL offers a significantly easier approach to control: specify the reward instead of the behaviour, finding an appropriate and implementable reward function have proven difficult. Specifying and finding a suitable reward function for RL problems is discussed next.

2.5.2 Reward learning

Reinforcement learning holds a major advantage over supervised learning when it comes to output specification: generating a reward signal requires less knowledge about the domain compared to specifying which actions to take. However, the reward function is a crucial ingredient that needs to be tuned in order for an RL agent to perform well (Sutton and Barto 2018). In particular, the episodic nature of robotics tasks makes reward sparse and difficult to learn from. Additionally, many robotic manipulation tasks are multi-objective. For example, folding a shirt often adheres to the subtasks discussed in Section 2.2: isolating the clothing piece, unfolding, folding, flattening and stacking. Therefore, the RL practitioner weights features that describe task performance. For example, in the peg-hole insertion task in (Vecerik et al. 2018), s the distance between the peg-socket and the peg-hole. Notably, the authors conclude that assigning weights to the features of the reward function is delicate and crucial for task performance. In practice, constructing reward function components and weighting them is often done by trial-and-error or a sweep over a predefined set of weights.

Learning reward functions is a viable alternative to manually specifying rewards as a way to avoid bias, overfitting and underspecification.

This process is known as reward shaping (Laud 2004): tuning the reward function based on empirical task performance. Hence, reward shaping is a method to embed prior task knowledge in the reward function in order to accelerate the learning process. However, even this trial-and-error process requires specifying which components constitute the reward function and how they are measured. For some domains, specifying and measuring the reward function components is non-trivial. A reward function for autonomous driving for example, should take collision with other objects, passenger comfort and in-lane driving into consideration. Cloth folding suffers from the same specification problem in addition to being hard to measure due to difficult state estimation caused by the deformations. Behavioural cloning is not a suitable alternative to learning the task from demonstration because it learns to copy the task execution instead of learning the task intent. Additionally, differences between the learning agent and the demonstrator's morphology make copying behaviour ambiguous.

Defining a reward function for the robotic folding task is difficult because it requires the state estimation of the cloth. Previous work (Doumanoglou et al. 2016; Miller et al. 2012) has extracted the contour of the textile using colour segmentation. In (Balaguer and Carpin 2011), a marked towel is tracked, allowing the calculation of the distance between points on the towel from the training sample and the example demonstrations so that it can be used as a reward for the agent. Their method requires prior information about the shape of the object in order to reconstruct the missing market points. More recent work in using (deep) reinforcement learning for robotic folding also used vision-based methods to define the reward function for the agent (Tsurumine et al. 2019; Matas, James, and Davison 2018). However, relying solely on visual inputs and marker clues does not scale well.

Another methodology to leverage example demonstrations is inverse reinforcement learning. Inverse RL is a category of algorithms that learn reward functions from demonstrations (Ng, Russell, et al. 2000). Formally, inverse RL can be defined using the MDP formalization of RL. RL tries to discover an optimal policy π^* by collecting rollouts $\mathscr{D} : \{\tau_i\} \sim \pi$ from the environment, with τ_i being a trajectory of state-actions pairs $\langle s_0, a_0, \cdots, s_T, a_T \rangle$, by maximizing the sum of expected rewards (cfr. Equation (2.13)). By contrast, inverse RL tries to discover the reward function $\mathscr{R}$ that explains the observed behaviour π in demonstrations $\mathscr{D}$. This learned reward function can then be used by the agent to learn policies. Inverse RL is challenging because it is an *underdefined problem*: there are multiple reward functions that can explain example behaviour. It is also difficult to evaluate the learned reward function because it requires training the agent until convergence. Many IRL algorithms assume optimal behaviour in the example demonstrations which is not always the case. Finally, in inverse reinforcement learning, there is an outer loop learning the reward function while the inner loop executes a learning procedure for finding an optimal policy given the current reward function. Recent methods have looked at integrating deep neural networks as a representation layer in inverse reinforcement learning (Finn, Levine, and Abbeel 2016; Ho and Ermon 2016; Fu, Luo, and Levine 2018). However, due to the two loops taking place, a lot of computational power is required for training. Speeding up the training process with a kinesthetic teach-in and updating instead of optimizing the reward function is explored in (Finn, Levine, and Abbeel 2016). Unfortunately, manually moving the end-effector of a robot proves to be unfeasible for difficult tasks like knot tying or folding clothing.

Ultimately, decoupling the dependency between reward learning and policy learning can speed up the process significantly, which is the topic explored in this work in Chapter 6. We focus in particular on harnessing the expressiveness of deep neural networks by using their latent space for constructing reward functions. The underlying hypothesis is that if the latent space contains relevant semantics, it can be used to extract reward functions by labelling data, using distance in embedding space or unsupervised learning. Labelling data is explored in (Singh et al. 2019): they construct a reward function based on an image classifier trained on successful goal states reached by teleoperating the robot towards the end state. A potential way for arriving at semantic-meaningful embedding that can be used for reward extraction is the self-supervised learning paradigm discussed in Section 2.4.2. Time, in particular, is a useful signal for self-supervision. This idea was pioneered in (Sermanet et al. 2018) in which they introduce TCN: generating contrasting examples based on the temporal dimensions in videos. TCNs uses multi-perspective video demonstrations as input and time as a supervisory signal. TCNs are demonstrated to learn meaningful semantic embeddings, which can be used for robotic pose imitation of humans. This is done by aligning video frames using nearest neighbours in embedding space. This is problematic in case a certain state machine or trajectory in embedding space has to be followed in order to solve the task. In (Dwibedi et al. 2018), TCNs are trained over multiple input frames such that the network is able to encode the position and velocity of objects in the scene. In (Hartikainen et al. 2019), time is used as a learned distance function for assigning environment rewards. However, their approach requires human intervention in order to select the desired goal states. (S. Nair et al. 2020) also uses time as a supervisory signal in videos of expert demonstrations to learn an optimal trajectory of states. However, they assume the possibility of visually removing the end-effector from the scene, which is not possible for all tasks. In the cloth folding example, it is not possible to stop and remove the end-effector in mid-air as the cloth would drop. (Sermanet et al. 2018) and other work (A. V. Nair et al. 2018) looks at expressing the reward function as the distance in latent space between the current state and the goal state. However, this is not possible when there is a trajectory in latent space that has to be followed in order to execute the task. For example, a cloth folding task can start the same way it ends: some pieces of cloth lie crumbled on the table while others are neatly folded and stacked on top of each other. Although TCNs are shown to be capable of robotic imitation of human poses, there is to the best of our knowledge no work that exclusively distills process monitoring metrics or reward functions from self-supervised representations trained on video demonstrations.

2.6 Datasets for robotic learning

The robotics community in general have accepted deep learning to be a powerful tool (Sünderhauf et al. 2018). This acceptance is evident from the surge in *deep learning* keywords in high-tier robotics conferences such as ICRA, the hosting of robotic application workshops at computer vision and machine learning conferences such as CVPR (Angelova et al. 2017) and NeurIPS (Posner et al. 2017) and the advent of a dedicated Conference on Robot Learning[7]. However, neural networks are known to be data-hungry due to their high parametrization. Additionally, RL requires many interactions with the environment to learn meaningful behaviour. Together with the high cost associated with collecting data on real robotic platforms, there is a need for datasets for robotic learning. This eminent need for high-quality datasets is also apparent in the general deep learning community with the launch of the NeurIPS 2021 Datasets and Benchmarks Track[8]. In addition to the availability of data, datasets need to uphold a high quality for successful training, generalizability and to avoid unwanted biases. A method to ensure quality is for example by standardizing the documentation process through datasheets (Gebru et al. 2018).

The availability of high-quality, real-life datasets enables to train robotic controllers that are unbiased, work on real problems and can be benchmarked.

The applications in robotic manipulation for datasets lie in applying supervised learning methods to train grasp quality predictors, identify grasping points and object states. Alternatively, offline RL, also known as batch RL and discussed in Section 2.5, can be employed to learn policies from gathered state-actions tuples without letting the policy interact with the environment itself. A successful approach to generate datasets for robotic manipulation is implemented in the Dex-Net (Mahler et al. 2017a) dataset containing 6.7 million synthetic robust grasps. Dex-Net offers RGB-D images with candidate grasping pose and grasping outcome of 1.500 object meshes. Generating synthetic grasping data is also explored by other authors (Depierre, Dellandréa, and Chen 2018; Redmon and Angelova 2015) but has the pitfall of potential transferability issues to the real world. This sim2real problem urges other authors to look at generating data in-vivo. However, human supervision for controlling robots is expensive and biased. Hence, a popular approach is to self-supervise the data collection process: use a heuristic motor control rule to execute grasps. In (Pinto and Gupta 2016), a dataset consisting of

7. http://www.robot-learning.org

8. https://blog.neurips.cc/2021/04/07/announcing-the-neurips-2021-datasets-and-benchmarks-track/

50.000 grasp attempts with a dual-armed robot is collected. Afterwards, a network is trained to predict a good grasping orientation from a given image patch. In (Levine et al. 2016), this approach is scaled-up. They generate two datasets containing 800.000 and 900.000 grasping attempts on two different clusters of robotic manipulators. They record images from a monocular RGB camera mounted over the shoulder of the arm. They synchronize the images with the delta end-effector pose and grasp the success outcome. Their dataset consists primarily of rigid objects, although some slightly deformable objects such as rubber ducks are present. This dataset is then employed for training a grasp quality predictor neural network. Robonet (Dasari et al. 2019) is a follow-up initiative that merges rollouts of 7 different robots from 4 different institutions in order to obtain 15 million frames containing variability across viewpoints, objects, robots and lighting conditions. Other researchers (Mandlekar et al. 2018) argue that self-supervised data collection is inefficient and prone to errors. This critique has led to RoboTurk (Mandlekar et al. 2018), an online platform that allows remotely operating robots for manipulation tasks. They collect 111 hours of RGB-D data matched with joint and effector sensor readings from 54 different robot operators for different tasks. One of the tasks consists of unfolding garments on a table and have been shown to be useful for training self-supervised embeddings.

Compared to rigid objects, the deformable object domain is less endowed with datasets. The largest cloth datasets are mainly constructed for automating retail applications such as clothing category identification, fashion landmark detection, image retrieval and recommendations systems. The first annotated cloth dataset is Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) which is a drop-in replacement for the MNIST dataset, often used for benchmarking ML algorithms. Fashion-MNIST contains 70.000 grayscale images of size 28×28 of fashion products from 10 categories. A modern variant of Fashion-MNIST is the DeepFashion (Liu et al. 2016) dataset, which contains 800.000 images scraped from the Google image search engine and from shopping websites. The dataset contains 8 different landmarks (i.e. collar, sleeve, waistline, and hem for each side), 46 clothing categories and 1.000 clothing attributes such as fabric material and fashion shape. This dataset is extended in (Ge et al. 2019) which provides more images, richer annotations and multiple clothing per frame to improve real-world realism. To reduce the noise introduced by scraping data, the FashionAI (X. Zou et al. 2019) dataset focuses on improving the automated annotation process by providing high-quality labels and a dedicated network architecture that assign attributes in a tree-like way. The attributes represent fashion semantics that are disassembled into hierarchical concepts. For example, a round collar bishop top is disentangled into “top look → sleeve region → style → cuff → bishop” and “top look → collar → round”. The neural network training uses a DAgger-like approach from behavioural cloning in which the network requests expert labels for cases in which it is uncertain. They collect 357.000 images of 6 categories of women’s clothing and 245 high-quality hierarchical structured annotations.

In contrast to annotated cloth data for retail applications, datasets for robotic learning of manipulation skills for clothing and deformable objects are not widely available. While a large body of research exists on cloth modelling and garment reconstruction from images (Bertiche, Madadi, and Escalera 2020; H. Zhu et al. 2020; T. Y. Wang et al. 2018), the availability of dedicated cloth folding data is scarce. To the best of our knowledge, the sole example containing a small set of deformable objects manipulations with a robot is in (Mandlekar et al. 2018). Other work concerning datasets for learning to manipulate cloth or other deformable object generate data in simulation, which is discussed in the following Section 2.7.

2.7 Simulation environments to accelerate learning

Physics simulators are a crucial tool for robotics researchers. The community often first tests hypotheses and methodologies in simulation and optionally transfers results to the real world. The main reason for using virtual platforms is their low cost, reproducibility and availability compared to real robots. Simulations can run faster than real-time, do not need an active operator and can provide the large dataset requirement for deep learning algorithms.

Simulation environments

Simulation environments are a virtual counterpart of the real physical environment that enables fast and save experimentation of robotic experiments but require transferring to the real world.

While a wide array of physics simulators can model and simulate a diverse set of phenomena, it is important for robotic researchers to have access to this physics engine within a robotic simulator. This requires the engine to support common robotic tools such as ROS integration for transfer to a real robotic platform, inverse kinematics, URDF import and models for joints, actuators and sensors. Another important but often forgotten feature is the ability for headless rendering. Running the simulation without having a physical display attached allows to use computation farms that often come without display. Common robotic simulators (Collins et al. 2021) such as Gazebo (Koenig and Howard 2004), MuJoCo (Todorov, Erez, and Tassa 2012) and PyBullet (Coumans and Bai 2016–2021) support robotic object manipulation tasks, however the deformable object simulation functionality is limited.

The limited support for cloth simulation in existing robotics simulators has led to many researchers implementing custom cloth simulators. Matas, James, and Davison (2018) for example extend the functionality of PyBullet to train a robot agent to fold cloth completely in simulation. DeformableRavens (Seita et al. 2021) and DEDO (Antonova et al. 2021) are other soft body simulation implemented in the PyBullet physics engine that allows simulating robotic interaction with ropes, fabrics and bags. Unfortunately, the visual and physical fidelity is significantly lower compared to other synthetically generated cloth data. SoftGym (X. Lin et al. 2020) provides simulated benchmarks for fluid, cloth and rope simulation. It uses Nvidia FleX particle system simulation, which is hardware-accelerated on GPU. Because SoftGym runs on the GPU, it allows doing more calculations in parallel, leading to physically and visually more realistic cloth behaviour.

However, the authors anticipate fundamental challenges when transferring policies trained on SoftGym to the real world. Finally, other relevant work examines to make the simulation itself differentiable. Differentiable physics simulation is a powerful technique that applies gradient-based methods to simulating physical systems. This way, it enables gradient-based optimization for control. Backpropagating gradients through a neural network controller and through the physical system has been shown to speed up the learning process for robot control tasks (Degrave et al. 2019). Extending this work to soft body simulations has been explored in (Liang, M. C. Lin, and Koltun 2019; Z. Huang et al. 2021) and have shown that gradient-based optimization method outperforms RL but fails on multi-stage tasks that require long-term planning.

2.7.1 Deformable object simulation methods

Forces applied to a deformable object both move the object and change the shape making high-fidelity modelling more difficult and computationally expensive compared to rigid object simulation. Deformable bodies require reasoning about the shape, dynamics and material properties of the object. Techniques in modelling cloth-like behaviour fall on a spectrum based on the computational budget allocated to the simulation. Offline simulation is on one side of the spectrum that prioritizes visual quality by means of physical realism. A prominent example can be found in the movie industry where GPU farms render multiple days for a two-hour video clip. In contrast, real-time systems are interactive and require running at a fixed frame rate. An important application of real-time simulation systems is found in video games. Games typically run between 30 Hz and 60 Hz leaving 15 ms and 30 ms computational budget per frame. Subtracting the time needed for core game features such as handling user input, game logic and AI leaves only a few milliseconds remaining for physical simulation. Unfortunately, reducing the resolution of the simulated object often leads to unsatisfying results (Müller et al. 2008). This is why real-time methods focus on reproducing the visual properties of physical processes. These constraints have driven the game industry to develop methods outside the offline physical simulation domain.

Physical simulation of deformable objects for a robotic learning environment requires characteristics of both real-time and offline simulation. For learning in simulation, we need physical realism for transfer to the real world and also needs to share resources for learning purposes and robot simulation. Offline cloth simulations are notoriously slow. For example ArcSim (Narain, Samii, and O'brien 2012), a realistic cloth simulator using FEM, requires 50 s to render a single frame of a fully dressed human character. Using these expensive simulations in a robotic learning environment would consume the computational budget, leaving little time for learning. To understand the rationale involved in selecting an appropriate simulator, we discuss the two major steps for implementing a deformable object simulation.

A first step in modelling deformable objects is choosing a shape representation. Choosing a representation changes the flexibility of the model and impacts the modelling of the dynamics. There are three major approaches for representing the shape of a deformable object:

1. First, implicit curves and surfaces can be used as a representation. These shapes are defined by an implicit equation $f : \mathbb{R}^n \to \mathbb{R}$ with $f(\mathbf{p}) = 0$ for which all points are on the surface. Implicit equation representations are primarily used in medical imaging in which level set methods are used for tracking deformable objects (Cremers 2006).

2. Parametric curves and surfaces, the second option for shape representation, are shapes controlled by a limited set of parameters. A 3D parametric surface is generated by a set of functions $\mathbf{q} : \mathbb{R}^2 \to \mathbb{R}^3$ with all cartesian points evaluated directly from their functional expression $\mathbf{q}(u, v) = \{x(u, v), y(u, v), z(u, v)\}^T$ where u and v are parameters. Splines and extensions such as B-splines and NURBS are parametric surfaces that can represent any type of deformable object. They allow a compact representation defined by control points that can move and deform the surface. However, depending on the

spline parametrization, moving control points can lead to erroneous deformations if for example the resulting curves do not lie on the convex hull of the control points. Learning in such a non-linear dynamical environment is difficult and a general solution for this is to date not available (Arriola-Rios et al. 2020).

3. A third possible shape representation is a mesh. A mesh stores the topology and geometry of an object represented by connected vertices. Meshes are the most used shape representation for a deformable object in simulation.

The second step in simulating deformable objects is choosing a dynamics model that will calculate how the chosen shape representation will deform on interaction with forces exerted on the object. In robotics, the important characteristics influencing the choice of dynamics model are computational cost, physical accuracy, visual fidelity and ease of use. In the following paragraphs, we consider two main approaches: FEM and particle-based methods.

The FEM is a widely used method for offline simulation of solid objects. FEM incorporates real physical material properties making simulations fairly realistic. This method generally works by reducing the general partial differential equations that describe the physical reality to systems of algebraic equations. Because these equations are often non-linear, solving the system in real-time becomes non-trivial. A way to speed up the solver is by linearizing the equations. This works for cases where deformations are small such as the analysis of buildings. However, the artefacts caused by linear approximations become significant in the case of cloth. This is why in real-time applications, particle-based methods are often desired above FEM. Particle simulations consist of atomic masses called particles that undergo forces such as gravity or a robot end-effector grasp. These forces drive particles to new positions solved by using Newton's second law of motion and a time integration scheme to solve the resulting differential equations. In the case of cloth, the particles are connected with their second-order neighbours and exert forces on each other in order to preserve the shape. These forces are modelled as springs, giving rise to a mass-spring system of particles. Compared to FEM, particle systems are faster to simulate and easier to implement. However, they are harder to tune and can be plagued by instabilities. In particular, choosing the right spring constant for the springs is notoriously difficult. Nonetheless, a spring-mass system approach for cloth simulation is a favourable approach as it conveniently handles the two-dimensional structure of cloth. This finding is also apparent in the robotic learning literature: modern research using cloth simulations implements a particle-based approach (Matas, James, and Davison 2018; Seita et al. 2021; Antonova et al. 2021; X. Lin et al. 2020). An exception is (Liang, M. C. Lin, and Koltun 2019) that utilizes FEM. We formally describe and implement a complete particle-based cloth simulation in Chapter 3.

2.7.2 Transferring simulation results to the real world

Simulation-based training methods allow for fast and safe experimentation for robotic tasks. However, differences between simulated physics and real-world physics exist due to undermodelling and errors in system parameter identification. This *reality gap* causes policies trained in simulation to suffer performance when deployed on the real platform. The resulting issue is labelled as the *Sim2Real* problem in robotics.

Sim2Real problem

The Sim2Real problem deals with overcoming undermodelling and system parameter identification errors in order to transfer policies trained in simulation to the real world.

The reality gap for training robotic controllers in simulation with deep RL became apparent in (F. Zhang et al. 2015), although the problem is not new. In the field of control theory, calibrating mathematical models of physical systems has been common practice (Ljung and Söderström 1983). This approach is known as *system identification* and consists of tuning the simulation parameters such that the simulated behaviour is as close as possible to the real behaviour. Control practitioners have calibrated models using local search techniques that search for the optimum by using a gradient-following technique (Ljung and Söderström 1983). Later, genetic algorithms have been shown to be competitive to deal with non-differentiable and non-linear search spaces (Kristinsson and Dumont 1992). Modern versions of system identification use machine learning methods (Chebotar et al. 2019) and differentiable physic simulation (Heiden et al. 2021) to learn complex dynamics, such as friction and contact parameters, from real data. A notable example in the deformable object manipulation domain is the early work of Howard and Bekey (2000) that deals with the system identification. In their work, a mass-spring-damper simulation of a deformable object is tuned to the real world by probing the object with the gripper containing a force sensor.

Another view on the reality gap is that it is an instance of a domain adaptation problem in which the distribution of the data of the source domain (i.e. the domain in which training is taking place) is being transformed to resemble the distribution of the data of the target domain (i.e. the test-time domain). There are three approaches to domain adaptation (Zhao, Queralta, and Westerlund 2020). (1) Discrepancy-based adaptation which measures feature distance between source and target domain and aligns their representation vector. (2) Adversarial-based adaptation that uses GANs as domain-invariant feature representation for training downstream neural network policies. (3) Reconstruction-based adaptation which learns domain-invariant features by having a reconstruction task. In deep RL for robotics, domain adaptation is used to improve the data efficiency of the work of (Levine et al. 2016) by introducing synthetic data. The synthetic and real data are used to train a GAN to transform synthetic images to real images (Bousmalis et al. 2018). These pseudo-real images reduce some of the reality gaps but operate at the pixel level and risk changing the semantics of the scene. Future work (Rao et al. 2020) alleviates this issue by introducing a cycle-consistency loss which encourages the preservation of semantics when inverse transforming the adapted image. Another approach is to learn a joint feature representation for source and target domain using GANs.James et al. (2019) for example, train a generator that transforms both simulated, randomized images and real-world images to a canonical representation which is used as a feature vector for policy training of a robotic grasping task.

Another category of Sim2Real methods is domain randomization. Domain randomization introduces variability in the elements constituting the simulation environment (Tobin et al. 2017). The hypothesis underlying domain randomization is that if the agent learns a policy successful in multiple variations of the same environment, then it should transfer to the real world which is just another permutation of all the variants seen during training. More formally, the goal of domain randomization is to find a parametrized policy π_θ that maximizes the expected reward over a distribution of simulation models ρ_ξ:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{\xi\sim\rho_\xi}\left[\mathbb{E}_{\tau\sim p(\tau|\pi_\theta,\xi)}[R(\tau)]\right], \tag{2.14}$$

where ξ represents a simulation configuration and $p(\tau|\pi_\theta,\xi)$ the likelihood of trajectory τ under policy π_θ and simulation configuration ξ. Simulation parameters can be categorized into visual randomizations and physical randomizations. Examples of simulation parameters are given in Table 2.1. Domain randomization has been successfully applied to transfer policies trained in simulation in a zero-shot manner to the real world (Tobin et al. 2017; Peng et al. 2018; Akkaya et al. 2019). However, domain randomization is known to require massive amounts of computational power. An alternative is guided domain randomization methods which uses domain randomization schedules, curricula (Raparthy et al. 2020), real-world calibration (Chebotar et al. 2019) or finds useful and informative randomizations (Mehta et al. 2019). Domain randomization is a popular approach for learning deformable object manipulation. For example, (Matas, James, and Davison 2018; Wu et al. 2020) transfer cloth manipulation tasks trained in simulation to the real world using domain randomization.

Finally, meta-learning can be considered as a Sim2Real method that takes the opposite approach to domain randomization. Instead of learning robust policies across a distribution of simulation environments, meta-learning learns a policy that is able to adapt rapidly to a changing environment and task.

Table 2.1 Non-exhaustive list of simulation parameters to vary in domain randomization.

Category	Simulation parameters
Visual	Object color, shape and position
Visual	Robot color, and position
Visual	Material texture
Visual	Lighting conditions
Visual	Camera pose
Physical	Object mass and friction
Physical	Robot morphology, mass and joint friction
Physical	Damping of physics integrator
Physical	Latency between action selection and execution
Physical	Observation noise

Domain randomization can then provide a task distribution for meta-learning.

2.8 Deformable object state perception through instrumentation

A typical approach for the robotic folding of textile relies on the use of vision in order to detect grasping points and to perform texture segmentation and pose estimation (Maitin-Shepard et al. 2010; Doumanoglou et al. 2016; Bersch, Pitzer, and Kammel 2011), as well as to estimate the state (Matas, James, and Davison 2018) and—in the case of reinforcement learning-based approaches—the reward function (Tsurumine et al. 2019). However, while vision is instrumental for recognizing and localizing objects, touch and force measurements become important once contact occurs, and the object is explored using the end-effector (Billard and Kragic 2019). Recent work (Tian et al. 2019) has illustrated this idea by applying a touch-based control method for a ball repositioning task, rolling dice and the deflection of a stick.

In line with other authors (Tian et al. 2019; M. A. Lee et al. 2019), we believe the use of tactile input and its fusion with other sensory information is crucial to learn complex robot manipulation tasks in an efficient way. We distinguish three functional locations where sensors can be added: instrumentation on the robot, on the target object and in the environment. We discuss these applications in the robotic manipulation context.

Instrumentation

Instrumentation is the process of applying sensors in the learning environment in order to accelerate learning.

A first location to instrument is on the robot itself. Vision-only sensors is the canonical approach to robotic manipulation task given the popularity of visual servoing. H. Lin et al. (2015) for example, pick up deformable objects by using a 3D laser scanner that discretizes objects into a tetrahedral mesh. In (Navarro-Alarcon et al. 2016), the shape of a volumetric deformable is adapted by quantifying the deformations using visual input. Tactile-only sensing has also been employed in, for example, (Drimus et al. 2014) that equipped a parallel gripper with a tactile array sensor to classify rigid and deformable objects. An example exploded view of an implementation is shown in Figure 2.10. Kappassov, Corrales, and Perdereau (2015) stresses the importance of having high-resolution tactile sensing in order to detect small variations in the object's shape. Similarly, Kuppuswamy, Alspach, et al. (2020) pioneer soft-bubble grippers: parallel air-filled finger membranes whose deformability allow both compliant grasping and sensing. The instrumentation of the finger with internal markers enable visuotactile sensing by tracking the optical flow pattern of the membrane deformation. They demonstrate how tactile information can be used for material and object classification. In (Kuppuswamy, Castro, et al. 2020), the soft-bubble gripper technology is used to estimate the pose of an object is by developing a contact patch tracking model. Yuen et al. (2017) develop sensory fabric sleeves that can be attached to end-effector joints for determining joint angles and end-effector position. (Jennifer C. Case et al. 2019; Jennifer C Case et al. 2018) implement a robotic skin containing sensors and actuators using stretchable, flexible substrates.

The sensory information of the robotic skin enables estimating physical properties such as the state and stiffness of the objects, which can be used for motor control tasks such a controlling the segments of a finger. Fusing multiple heterogenous sensor sources for deformable object manipulation is proposed in (Khalil, Payeur, and Cretu 2010). This setup uses binary tactile pads and a camera for contour tracking of deformable objects. Frank et al. (2010) estimate physical properties for simulating deformable objects using multi-perspective visual and tactile information. Multiple modalities can also be incorporated in the learning frameworks discussed in Section 2.3. For a survey on using deep neural networks for multimodal learning, we refer to (Ramachandram and Taylor 2017). In the context of object manipulation, (M. A. Lee et al. 2020) learn a self-supervised representation of pixels, force sensing and proprioception for peg insertion task. Each input modality is processed through a dedicated architecture; a CNN for the pixel input, a CNN with causal convolutions similar to WaveNet (van den Oord et al. 2016) and a fully connected neural network for proprioception and sensor readings. Each of these architectures produces a feature vector which is merged into a single feature vector using a Product of Experts (Hinton 2002). Their ablation study reveals that all modalities are used as the system performance drops significantly when removing input streams. Similar approaches and conclusions have been found in (Calandra et al. 2018; Droniou, Ivaldi, and Sigaud 2015; Balakuntala et al. 2021). Zambelli and Demirisy (2016) adds a microphone in addition to cameras and tactile cells for a humanoid robot to play the piano. They find that multimodal sensing is crucial for a multimodal task such as piano playing.

A second location where instrumentation is applied is on the target object to manipulate itself. A prominent example is the instrumented Rubik's cube in (Akkaya et al. 2019). To avoid letting the robot learn to infer the Rubik's cube state from the camera, they equip the cube with rotary encoders to track face rotations. In the deformable object manipulation domain, Bersch, Pitzer, and Kammel (2011) and Elbrechter, Haschke, and Ritter (2012) apply fiducial markers on cloth and paper for state estimation. However, visual markers suffer from occlusion and make it hard

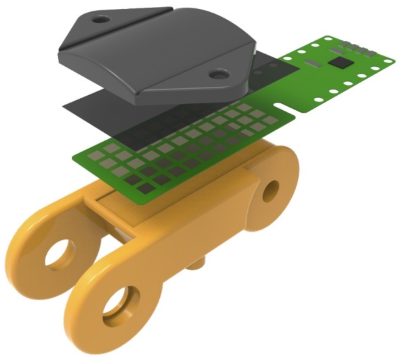


Figure 2.10 Example of instrumenting a robot's finger with tactile sensors. A grid-like sensor array is attached to the finger and protected by silicone pads.

to generalize for other clothing articles without markers. Smart textiles, i.e. textiles that are able to sense stimuli from the environment and react accordingly (Schneegass and Amft 2017), are being used in numerous applications such as health monitoring (Cochrane, Hertleer, and Schwarz-Pfeiffer 2016), sports (Lo Presti et al. 2019) and robotics (Yuen et al. 2017). An example of textile with integrated electronics is shown in Figure 2.11. Textile able to infer its own state is of particular interest for learning to manipulate clothing articles. To the best of our knowledge, there is no work of physical smart textiles for learning deformable object manipulation with robots.

A final entity to instrument are elements found in the environment. For example, a table can be equipped with force sensors that provide the location of objects to a robot. A concrete example can be found in (Kimura et al. 2013) who place a scale on the table with objects that have to be recognized and add the weight as an input modality for control tasks.

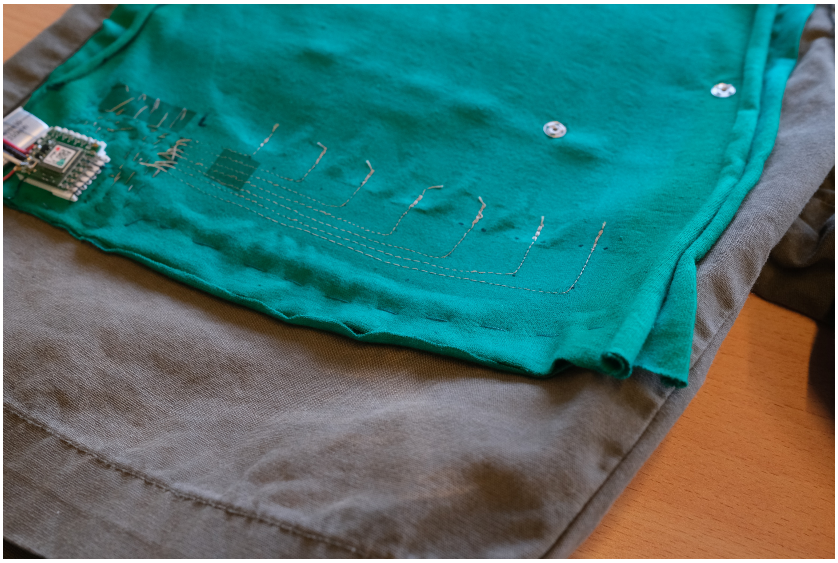


Figure 2.11 Example of a smart textile implementation using a piezoresistive polymer attached inside the fabric.

2.9 Conclusion

This chapter has reviewed the transition from traditional control pipelines for deformable object manipulation to leveraging learning methods. We have discussed methodologies, tools and approaches to building a cloth folding pipeline while identifying strengths, weaknesses and gaps. One of those gaps is the state estimation of cloth in the modern deep learning era. We found that the majority of work employ vision for estimating the state of clothing articles. In our research, we step away from this vast focus on vision-only sensing and integrate tactile sensors in order to create a smart cloth that can tell a robot its state. We stressed the importance of simulation environments for robotic learning and the importance of realistic cloth simulation. We will examine a basic implementation and discuss its usefulness, Sim2Real pitfalls and application for learning to fold clothing. We highlighted the potential for example-based learning and learning from interaction using expressive representation methods in deep learning. In this research, we combine both approaches to exploit their strengths: we use human demonstrations as examples for distilling task intent while letting the robot interact with the environment. To avoid expensive data labelling, we will introduce a self-supervised approach. We will start introducing our research in the next chapter that explores the use of simulation for robotic manipulation of deformable objects.

3

3

Robotic folding in simulation

We simulate cloth on GPU to
train a robot to fold cloth.

3 Robotic folding in simulation

The previous chapter discussed that simulators are omnipresent in the robotics community. In this chapter, we zoom in on this observation by exploring the use of simulation for training robotic controllers for deformable object manipulation. First, we define the components that make up a robotics simulator. Then, we compare different simulation technologies and finally discuss results on learning to fold in simulation.

3.1 Digital twins

Using a digital representation of a robot and the environment in which it operates allows reducing costly time on the real robot. This use-case has given rise to the term *digital twin* meaning that the physical platform has a virtual representation which can be utilized. This digital twin is a cost-effective tool for learning and safe experimentation. These benefits have proven their merit in the robotics field which has led to a plethora of simulator choices available to researchers (Collins et al. 2021). However, navigating the simulation landscape is difficult and requires pinpointing exact requirements. For researching the use of simulation for learning to fold cloth, we discuss the following requirements in this section:

- The simulation tool needs to be able to simulate robots and the associated physics.
- In addition to the rigid body simulation, our research requires simulating cloth dynamics.

- To bridge the virtual environment to the physical world, the simulation requires functionality to communicate with the physical robot.
- Machine learning and reinforcement learning libraries often require a Python interpreter. Hence, the simulation environment should be able to be interfaced with from a Python runtime environment.

3.2 Robotic simulation

A robotic simulator encapsulates a physics simulator while exposing other functionalities specific to solving tasks with robots. The physics simulation, also called physics engine, provides realistic modelling and simulation of physical phenomena. This includes handling the construction and articulation of kinematic chains, querying collision detection, providing friction models and optionally has built-in soft body support. The distinction between physics and robotic simulations helps understanding the simulators landscape as some physics engines have developed into robotics engines and can be found as physics backend in other robotic simulators. In Section 2.7, we discussed two categories of physics engines: real-time and offline. For robotic learning purposes, we prioritize simulation speed above accuracy so we only consider real-time physic engines. There is a wide variety of real-time physics simulators available such as Bullet (Coumans 2015), PhysX (Nvidia 2021), Havok (Havok 2021), ODE (Smith 2001) and MuJoCo (Todorov, Erez, and Tassa 2012). With the exception of MuJoCo, these physics engines are primarily developed for gaming purposes that require real-time simulation. However, in gaming and modelling applications, most bodies have few or even no joints or constraints. Consequently, many physics engines model physical bodies with a Cartesian representation in which each rigid body has 6 degrees of freedom. The joints of a body then become constraints imposed on the $6N$-dimensional space. This is in contrast to robots that are multibody systems of N constrained links that makes the system dimensionality much closer to N instead of $6N$. Hence, we have a strong preference for robotic simulators that have underlying physic engines using generalized joint coordinates that provide better stability, speed and accuracy. The difference between generalized coordinates versus Cartesian coordinates representation is illustrated in Figure 3.1. This example demonstrates that an inverted double pendulum can be represented as a 12 degrees of freedom system with 10 constraints or as a 2 degrees of freedom system without constraints.

On top of integrating a physics simulation, the robotic simulator exposes functionality specific to the robotics domain. One of these features is importing predescriped robot models that are often expressed in the Unified Robot Description Format[1]. The simulator must provide actuator models for position control, velocity control, and torque control in order to control the physical arms. Forward kinematics and inverse kinematics are needed for path planning functionality. Typically, robots for manipulation tasks are equipped with various sensors such as RGB-D cameras, torque and force sensors which need to be supported by the simulator as well. Similarly to RGB-D cameras, the simulator benefits from having a rendering pipeline for visualization and debugging purposes. This rendering pipeline can be used as RGB-D camera and is preferably able to execute headless for server-side rendering. In order for the digital twin to communicate with its physical counterpart, it needs messaging channels to the real platform. The communication is often provided through ROS[2]; a middleware software suit providing hardware abstraction through message-passing nodes. Finally, we want the robotic simulator to interface with Python interpreters given that the large bulk of machine learning research and libraries are written in Python. Python bindings also allows us to circumvent the explicit need of a robotics simulator with ROS integration as the robot used in this research has direct communication channels in Python.

1. http://wiki.ros.org/urdf
2. https://www.ros.org

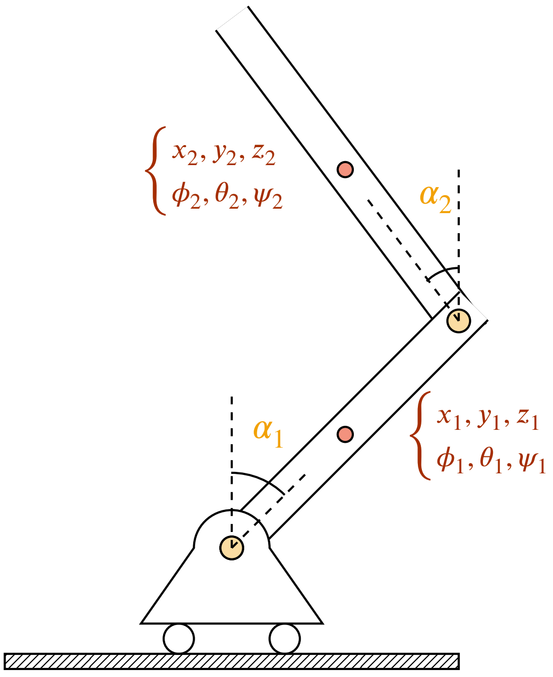


Figure 3.1 Comparison between generalized and Cartesian coordinates for representing multibody kinematic chains. An inverted double pendulum can be represented as two links each having 6 degrees of freedom: 3 possible translations and 3 possible rotations. The rotations ϕ_i, θ_i, ψ_i refer to roll, pitch, yaw respectively. In this Cartesian representation, indicated in red, all translations and 2 rotations are constrained per link. The joint coordinate representation, indicated in yellow, on the other hand does not have to impose any constraints because it represents the system with 2 degrees of freedom α_1 and α_2.

3.2.1 Qualitative comparison of popular robot simulation technologies

Performing an in-depth, exhaustive comparison between popular robotics simulators for manipulation tasks is non-trivial. A thorough comparison would require setting up multiple scenarios, relevant metrics and experience with all the considered simulation technologies. For example, it is unclear whether we should compare accuracy, scalability, stability or speed. Furthermore, it is hard to concretely quantify such metrics. For example, accuracy is difficult to quantify in the absence of analytical solutions. Another metric we could consider is performance in the context of the robotic manipulation tasks. However, the task performance also depends heavily on the used controller. Giovanni and Yin (2011) for example, found that the simulation engine does not matter when using robust controllers for generating gaits. Given that all robotic simulators at the time did not provide adequate cloth simulation support, we instead performed a qualitative tradeoff, consulted the documentation, forums and the limited amount of literature available at the time comparing the simulators (Staranowicz and Mariottini 2011; Erez, Tassa, and Todorov 2015). We compare the following robotic simulators next: PyBullet, MuJoCo, Gazebo and Unity. Other notable simulators like Taichi (Hu et al. 2019), Isaac[3] and ThreeDWorld (Gan et al. 2021) are excluded from the current comparison due to not being available or just being released at time of implementing our simulation. A summary of the simulators we considered can be found in Table 3.1.

PyBullet is a robotics framework written in the Python programming language on top of the Bullet physics engine. It provides robot functionality with Python bindings. PyBullet has a focus on machine learning in robotics. At time of executing this research, PyBullet was recently introduced and a one-man effort making the implementation immature. Many features needed yet to be implemented or were not exposed to the Python API. In addition, Bullet recently switched from Cartesian to joint coordinates rep-

3. https://developer.nvidia.com/isaac-sdk

resentation which was not thoroughly debugged. Compared to main-stream game engines, the rendering capabilities of PyBullet are subpar. Finally, PyBullet provides an experimental soft body implementation which we, among other authors (Matas, James, and Davison 2018; Seita et al. 2021), found unusable. The soft body physics caused cloth to tunnel or explode on grasping attempts and only wireframe rendering was possible at the time. PyBullet has been used for learning to fold cloth (Matas, James, and Davison 2018), learning via virtual reality demonstrations (Mahjourian et al. 2019) and learning quadruped locomotion with Sim2Real transfer (Tan et al. 2018).

Mujoco® is a general purpose physics engine developed specifically for robotics research. MuJoCo is generally known for its stable and efficient multibody system dynamics (Erez, Tassa, and Todorov 2015) and has been used for learning robotic manipulation (Rajeswaran et al. 2017), dexterous manipulation (Akkaya et al. 2019) and locomotion (Heess et al. 2017). Most of our required features are supported by MuJoCo with the notable exception of inverse kinematics and path planning. At time of executing our research, MuJoCo required a paid license which made it unfit for our purposes to be able to run simulations on remote servers[4]. MuJoCo provides volumetric soft body simulation of which the generalization towards planar soft bodies is unclear.

Gazebo is an open-source robotics simulation environment supported by ROS. It exposes multiple physics engines, most notably Bullet and ODE by default. The Bullet physics engine suffers from the shortcomings we mentioned while ODE only offers Cartesian multibody representation, making it slow and potentially unstable for robotics. Gazebo is strongly integrated with the ROS ecosystem, making it very feature-complete for transfer to real robots. None of the physics engines underpinning Gazebo offer decent cloth simulation. Gazebo is widely used in the robotics community, for example autonomous navigation (Imanberdiyev et al. 2016) and visual servoing (Shi et al. 2018).

Unity® is a game engine providing a software development environment and tools for game development. With the Unity ML-

4. MuJoCo was made freely available in October 2021.

Agents toolkit[5]and Unity Robotics Hub[6], Unity is laying down a strong footing in the robotics machine learning community. The Unity Robotics Hub enables pulling in functionality from the ROS ecosystem while the Unity ML-Agents toolkit provides a Python API to the underlying simulation environment. Additionally, Unity has integrated the reduced articulation functionality from the PhysX physics engine[7] making multibody simulation like robots physically more accurate with minimal joint errors. The rendering capabilities of Unity are of considerably higher fidelity compared to the previously discussed simulators. The high fidelity is due to the photorealistic rendering capabilities. Unity is being used by AAA game development studios and popular among solo game developers due to the rapid software prototyping tools it provides. Unity has been used for robotic manipulation of soft tissues (Tagliabue et al. 2020) and as backend rendering engine for learning dexterous manipulation (Akkaya et al. 2019).

Table 3.1 Qualitative comparison of the robotic simulation technologies considered for learning to fold clothing.

Requirement	PyBullet	MuJoCo	Gazebo	Unity
Stable physics	√	√	√	√
URDF support	√	√	√	√
Minimal coordinates	±*	√	±*	√
Python bindings	√	±†	√	√
High-fidelity rendering	—	—	—	√
Soft body support	±*	±‡	±*	±*
Inverse kinematics	√	±§	√	±§

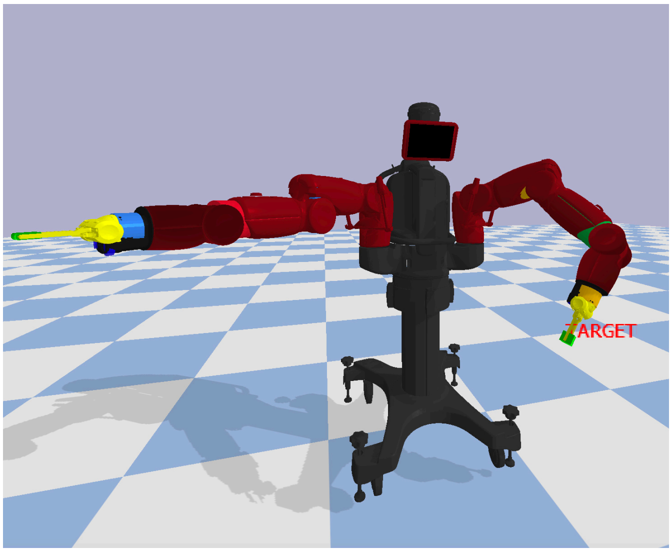
* Supported but immature
† Via community initiatives
‡ Mature but limited features
§ Indirectly, via ROS

5. https://github.com/Unity-Technologies/ml-agents
6. https://github.com/Unity-Technologies/Unity-Robotics-Hub
7. https://gameworksdocs.nvidia.com/PhysX/4.0/documentation/PhysXGuide/Manual/Articulations.html#reduced-coordinate-articulations

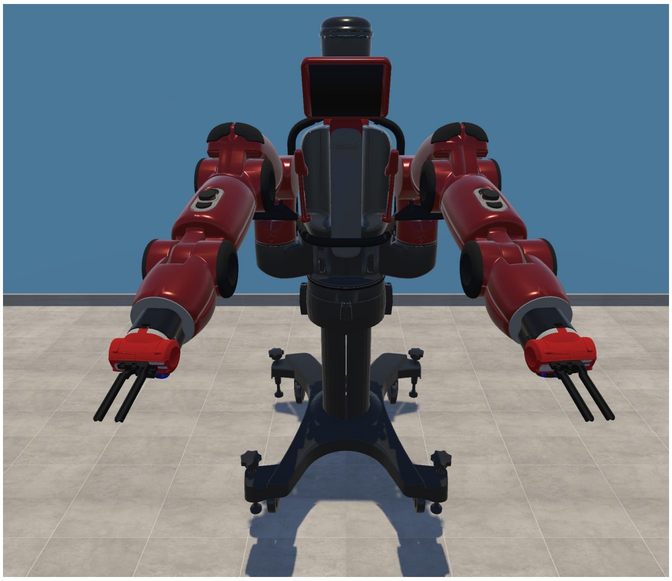
Ultimately, we decide to use Unity as robotics research tool. This decision is rationalized by Unity covering most of our required features as shown in Table 3.1. Notably their recent addition of supporting a physics solver for minimal joint coordinate multi-bodies with the high definition render pipeline sets Unity apart from other simulation technologies. Unity demonstrates to be incumbent player in robotics by dedicating software teams to their ML and robotics department. Compared to dedicated robotics simulators, Unity produces much higher visual fidelity due to photorealistic rendering. For example, we compare the visuals of a simulated Baxter robot in PyBullet and Unity in Figure 3.2. In addition, the Unity IDE is a professional and productive working environment. The community is known to be supportive and provides tools for other developers to use. This allows rapid prototyping of new features and experiments. Rapid prototyping together with the above discussed functionality makes Unity a suitable candidate for PhD researchers that are not backed by a large development team.

3.3 Cloth simulation using particle systems

Choosing a cloth simulation method requires trading off the geometrical resolution, computational speed, robustness and realism. The *geometrical* complexity needs to be at a resolution that can accurately represent small deformations and wrinkles that typically appear in clothing items. The *speed* of the simulation is important for generating massive datasets, which is required when data-hungry learning algorithms are used such as deep neural networks. The computational speed depends on the chosen simulation algorithm, temporal and geometrical resolution. With *robustness*, we cover requirements such as consistency when repeating the same cloth manipulations and stability when large forces are exerted. For example, game engines apply damping in order to avoid implosion of the forces and corresponding displacements. Finally, *realism* refers to the realism of the physical behaviour and visual quality. In this regard, we distinguish visual plausibility versus physical plausibility. The former refers to the



(a) Baxter robot visualization in PyBullet



(b) Baxter robot visualization in Unity

Figure 3.2 Simulated Baxter: PyBullet vs Unity. The difference in rendering capabilities can be seen in, for example, Unity being able to sample texture images while Bullet uses solid colours. Unity also allows defining the material's roughness; in Unity there is a black, plastic casing around the elbow leading to a complete diffuse light reflection. In contrast, there is a visible specular reflection on elbow joint of Baxter in Bullet.

rendering quality while the latter indicates how realistic the simulated behaviour is. Achieving realistic visualization is possible with modern rendering technologies such as physically based rendering methods that use the principles of physics to model the interaction of light and materials. Contrarily, modelling and tuning physical parameters of a simulation to achieve physical behaviour that follows physics in the real world completely, is difficult. This problem is enlarged in the case of high degree of freedom objects such as clothing articles. Hence, in this regard we prioritize visual fidelity above physical fidelity. Moreover, when using a simulation method, one can tune the degree of realism and geometrical resolution in order to achieve quicker simulation times. In heart surgery simulators for example, it is known that the simulated deformations of tissues do not need to be physically accurate. Instead, the deformations should be consistent and *look* physically realistic (Bro-Nielsen 1998) in order to facilitate the suspension of disbelief.

Modelling consistent and seemingly realistic deformations is the primary challenge for cloth simulation. For example, if we consider cloth to be constituted of n elementary particles, then computation of the forces on a particle requires incorporating pairwise interactions from all other particles leading to a $\mathcal{O}(n^2)$ computation. This computation is challenging to do in real-time given that cloth often uses many vertices. For example, simulating the dress shown in Figure 3.3, requires calculating forces for the 17244 vertices leading to a computation of $\mathcal{O}(n^2)$ with $n = 17244$.

3.3.1 Representing cloth with particle systems

In Section 2.7.1 we discussed two popular approaches for soft body simulations: particle systems and FEM. To recapitulate; FEM is a method to solve continuous equations that govern the behaviour of soft body dynamics by discretizing and linearising the geometry and material properties. The main advantage of FEM is physical realism: we calculate approximate solutions to the actual equations governing deformations based on elasticity theory. However, the main problem with FEM is simulation

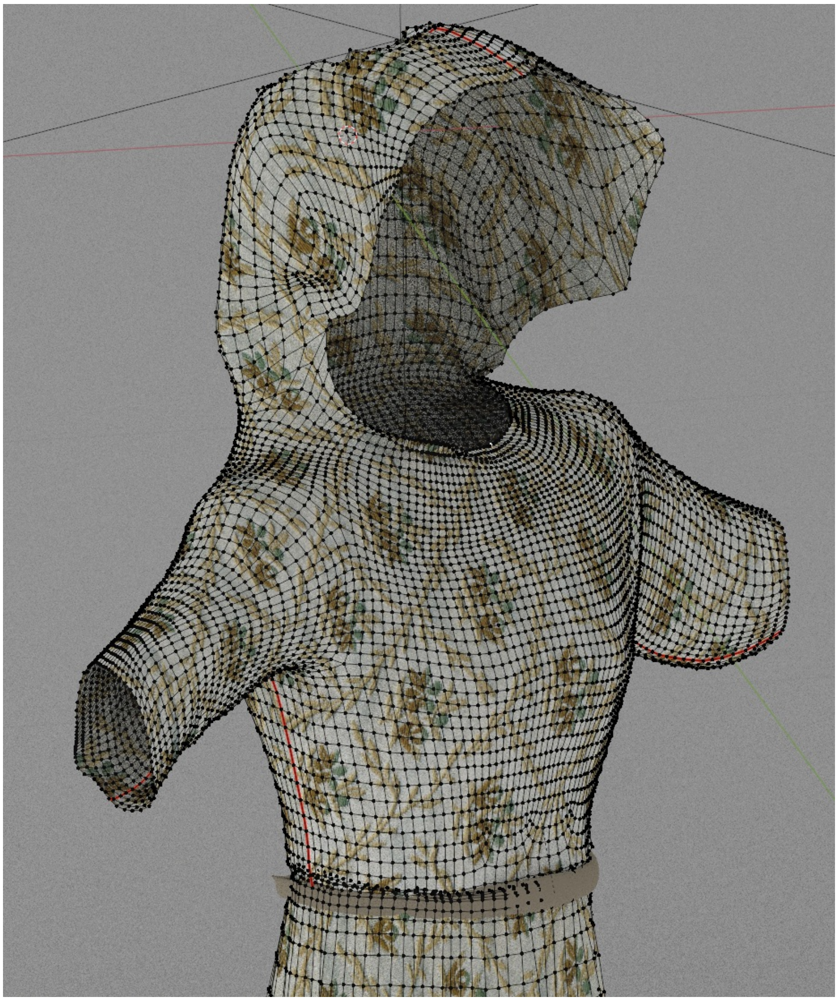


Figure 3.3 Example of a close-up of 3D modelled dress. The vertices and edges are displayed on top of the mesh. The cloth model and textures were designed by user *mnphmnmn* on turbosquid: https://www.turbosquid.com/Search/Artists/mnphmnmn.

speed: each simulation step requires solving a system of equations which is a computationally expensive operation. On the contrary, particle systems combine basic physics and mathematical constraints in order to control the dynamic behaviour of objects. The formulation of particle systems additionally allows to easily exploit dedicated hardware like GPUs. For these reasons, we employ a particle-based approach to cloth simulation in this research. We describe particle systems for simulating cloth next.

The computer graphics community has taken a different approach for modelling natural phenomena such as fire, water, foliage, smoke and deformable objects compared to standard computer image synthesis techniques for rigid objects. This is due to the continuously changing, irregular shapes and positions of the elements that make up these phenomena. A well-established method to model such systems is representing each modular atom of the system by a point mass, labelled particle, whose position and velocity are governed by physical laws. This procedural method falls under the category of particle systems. When organizing particles in a fixed topology and connecting neighbouring particles with springs that govern the shape deformation, a spring-mass system is formed.

A cloth $\mathscr{C}$ is constructed by bundling a collection of connected particles $i \in \mathscr{C}$. An example mesh is demonstrated in Figure 3.4. Each particle contains mass and inertia but does not take up volume. A particle i is defined by a 3D position $\mathbf{x}_i \in \mathbb{R}^3$ and velocity $\mathbf{v}_i \in \mathbb{R}^3$. We can organize all positions and velocities of the cloth $\mathscr{C}$ with N particles in a vector data structure:

$$\mathbf{x} = \begin{bmatrix} x_x^{(0)} \\ x_y^{(0)} \\ x_z^{(0)} \\ \vdots \\ x_x^{(N-1)} \\ x_y^{(N-1)} \\ x_z^{(N-1)} \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_x^{(0)} \\ v_y^{(0)} \\ v_z^{(0)} \\ \vdots \\ v_x^{(N-1)} \\ v_y^{(N-1)} \\ v_z^{(N-1)} \end{bmatrix}. \tag{3.1}$$

The position $\mathbf{x}$ and velocity vector $\mathbf{v}$ make up the state of the par-

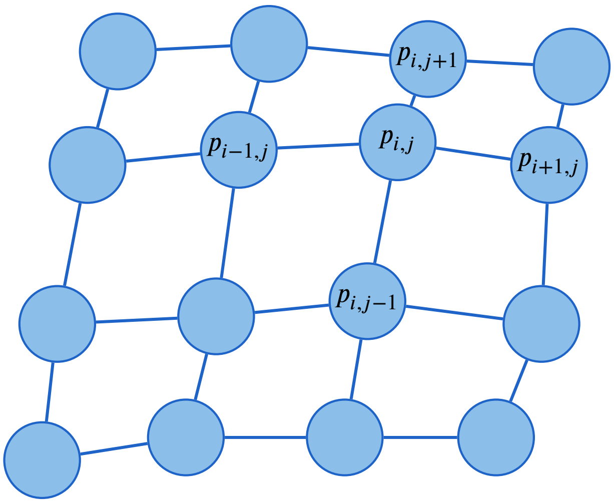


Figure 3.4 A mesh of connected particles. A particle $p_{i,j}$ is located in row i and column j of the mesh. In this example, each particle is connected only to its first-order neighbourhood.

ticle system or cloth $\mathscr{C}$. The individual positions and velocities are indexed by $v_x^{(i)}$ in which superscript (i) indicates particle i and subscript x refers to the x-component of the position and velocity vector. The positions $\mathbf{x}$ and velocities $\mathbf{v}$ will change over time based on internal spring forces in the cloth and external forces like gravity and interaction with the robot and environment. These forces obey Newton's second law of motion:

$$\mathbf{f} = m\mathbf{a}, \tag{3.2}$$

where $\mathbf{f}$ is force, m the mass and $\mathbf{a}$ acceleration; the first derivative of velocity $\mathbf{v}$ or the second derivative of position $\mathbf{x}$.

A spring connects two particles. The types of springs we employ will determine the topology and physics behaviour of the cloth. We elaborate in Section 3.3.3 the type of springs we use for achieving cloth-like behaviour. The springs follow Hook's law: the force exhibited by the spring on the connecting particles is linear in the displacement from the resting distance of the spring. Concretely, the force $\mathbf{s}_{ij}$ exhibited by the spring connecting particles i and j is given by:

$$\mathbf{s}_{ij} = k_{ij}\left(l_{ij} - \left\|\mathbf{d}_{ij}\right\|\right)\frac{\mathbf{d}_{ij}}{\left\|\mathbf{d}_{ij}\right\|}. \tag{3.3}$$

In this equation, k_{ij} is the spring stiffness of the spring connecting particle i and particle j. Similarly, l_{ij} is the resting distance of this particular spring. $\mathbf{d}_{ij}$ represents the distance between particles i and j: $\mathbf{d}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. Equation (3.3) is negative when the particles move away and stretch the spring past its resting distance. In this case, the spring will exert a negative force to attract the particles back together. Similarly, the spring will repel the particles when they are closer together than the nominal distance.

Incorporating external forces $\mathbf{f}_{\text{ext}} = \sum_e \mathbf{f}_e$ working on the particle i, the dynamics of the particle is given by filling in Equation (3.2):

$$m_i\mathbf{a}_i = \sum_j \mathbf{s}_{ij} + \mathbf{f}_{\text{ext}}. \tag{3.4}$$

Determining how to calculate the new positions based on the dynamics Equation (3.4), is discussed next.

3.3.2 Advancing the particle simulation

Differential equations arise when simulating physics. This is a consequence of finding new positions based on the forces working on the simulated objects. Common numerical methods to integrate the differential equations linearize the equations around a small time step in order to arrive at new positions of the objects. Practically, this means we have to translate the net force on each particle to an acceleration vector. Using this acceleration vector, we can calculate new velocities for the next time step. We can then update the positions by moving the particles according to the new velocity vectors. Concretely, to solve the second order differential equation given in Equation (3.4), one can express the equation as two first order differential equations that can be solved with standard methods, for example, Euler integration or Runga Kutta. Euler integration for example uses following update rule based on the Taylor series expansion:

$$\begin{aligned} \mathbf{x}(t + \Delta t) &= \mathbf{x}(t) + \Delta t \cdot \mathbf{v}(t) \\ \mathbf{v}(t + \Delta t) &= \mathbf{v}(t) + \Delta t \cdot (\mathbf{f}_{\text{ext}}/m). \end{aligned} \tag{3.5}$$

This type of Euler integration is often denoted as *explicit Euler* integration to contrast with the *implicit Euler* or backward euler integration method that uses quantities of next time step. Although explicit Euler integration is simple to compute, it is inherently unstable because of the linearization. By assuming the force is constant during the timestep Δt, the simulation can overshoot, gain energy and eventually explode. We can reduce this risk by minimizing the timestep Δt, often denoted delta time. However, reducing Δt leads to longer simulation time.

In our work, we use a different integration scheme called **Verlet integration**, popularized in (Jakobsen 2001) for real-time cloth simulation. Verlet integration combines one forward and one backward third-order Taylor expansion of the positions $\mathbf{x}(t)$ leading to the following update rule:

$$\begin{aligned} \mathbf{x}(t + \Delta t) &= \mathbf{x}(t) + (1 - \alpha)\mathbf{v}(t)\Delta t + \frac{\mathbf{f}_{\text{ext}}}{m}(\Delta t)^2 \\ \mathbf{v}(t + \Delta t) &= \frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t}. \end{aligned} \tag{3.6}$$

Similarly to the update rule discussed in Equation (2.1) for updating the weights of a neural network, we apply damping by weighting the old and the new positions using the damping factor α. This reduces the energy in the simulation. Equation (3.6) shows that Verlet integration operates on positions only, which is compatible when using position based dynamics (Müller et al. 2007). Position based dynamics directly manipulates the position of particles instead of integrating forces. The benefit of this is that we avoid choosing the spring constants k_{ij} in Equation (3.3). This formulation also allows handling collisions straightforward: instead of applying forces to resolve penetration, one can project the objects to valid locations.

Note that at this point, we have discretized the cloth $\mathscr{C}$ in two dimensions:

1. Discretization in space: a continuous cloth is represented by interconnected particles.
2. Discretization in time: continuous time will be divided into discrete time steps of size Δt to advance the physics in the simulation.

3.3.3 Topology constraints

The springs introduced in Section 3.3.1 function as constraints and flow of energy in the particle system. Hence we will use the terms spring and constraint interchangeably. The topology of these connections will determine the global behaviour of the cloth. In Figure 3.5, we create a rectangular grid with particles that are evenly spaced. We connect each particle with its second-order neighbourhood which leads to three types of springs (Provot 1995). The first type of springs are the *structural springs*, drawn in blue, that resist stretching the lattice. The shear springs, indicated in red, connect diagonal neighbours and resist shearing forces. This prevents the cloth from collapsing inwards. The last spring type, the bending spring drawn in yellow, connect the second order horizontal and vertical neighbours. This type of spring allows resisting bending for local patches when the resolution is high.

3.3.4 Collision

Collision handling in the cloth simulation is split into two different types:

- Cloth-object collision: collisions between rigid object and the cloth.
- Cloth-cloth collision: intercollision of the cloth with itself.

Conceptually, we split the collision handling into two phases: (1) collision detection phase and (2) collision response phase. First, we find the colliding parts and second, we resolve the collision to make sure objects do not penetrate.

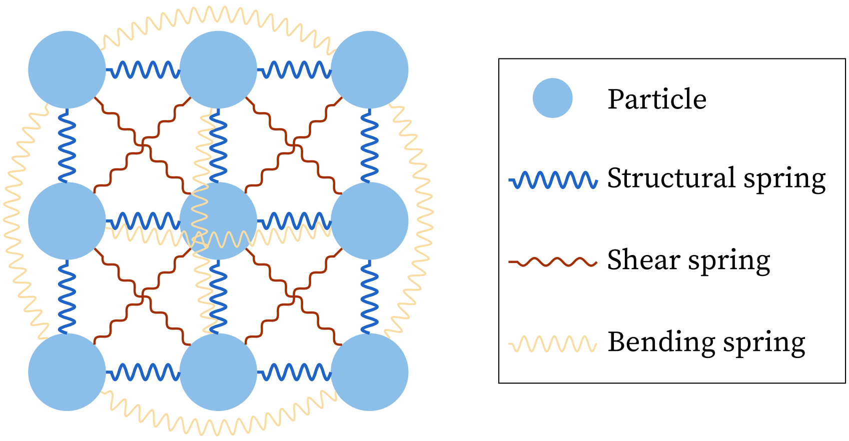


Figure 3.5 Mass-spring particle system: a rectangular grid of particles connected by different types of springs.

We consider the following types of collisions in our simulation that require detection and resolving:

- For the *cloth-table* collision, we represent the top of the table as a plane. Collision can then be detected when any particle position passes the plane. Collision is resolved by projecting the penetrating particle back to the surface.
- *Cloth-sphere* and *cloth-cuboid* collision is detected by checking whether every particle lies inside the sphere or cuboid.
- To implement *cloth-cloth* collision, we utilize a brute-force approach in which we compare every particle on the mesh with every other particle. This exhaustive comparison is feasible because we implement the simulation on GPU that allows checking every particle in one pass.
- *Cloth-gripper* contact modelling is simplified considerably by assuming the end-effector snaps the cloth once it is in close proximity. We utilize this magnetic-like approach to avoid complex contact modelling.

3.3.5 Rendering

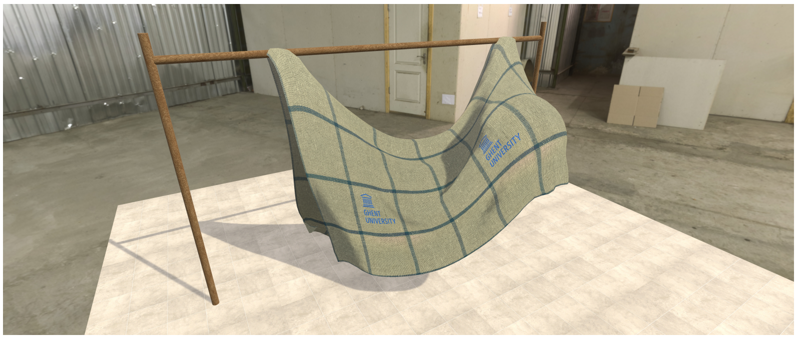
Although we stated that a particle is a volumeless entity, we can choose how to represent the point mass. The idea is that if we know the location of a particle, we can place the desired graphical representation at that location. We visualize the cloth by placing vertices at the locations of the particles. Then, we triangulate by connecting east and south neighbours. We calculate the normal vector on this triangle for shading using Blinn-Phong shading (Blinn 1977). Texturing is done by sampling cloth texture images.

3.3.6 Implementation results

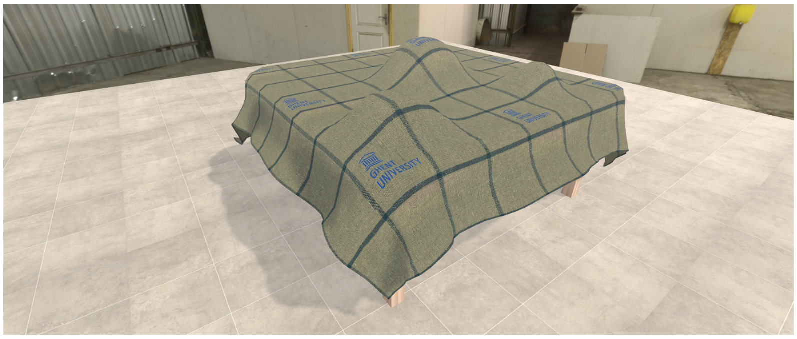
We implement the cloth simulation described in this section using Unity[8]. In order to speed up calculations, we implement the simulation on GPU using shaders. We advance the simulation using a fixed delta time of 0.02 s for all physics steps in order to have deterministic physics. We subdivide each physics step in multiple cloth physics step in order to have more accurate cloth physics.

Figure 3.6 shows the resulting cloth. The cloth mesh can be generated procedurally as for example the cloth in Figure 3.6a or from an existing mesh as shown in Figure 3.6c. Translating arbitrary meshes to mass-spring systems is done by clustering the mesh vertices and applying springs based on the mesh tangents. The rectangular mesh has a resolution of 100×100 particles and achieves 250 fps. We verify the stability of the simulation by moving particles around and penetrating the mesh with cuboids and spheres.

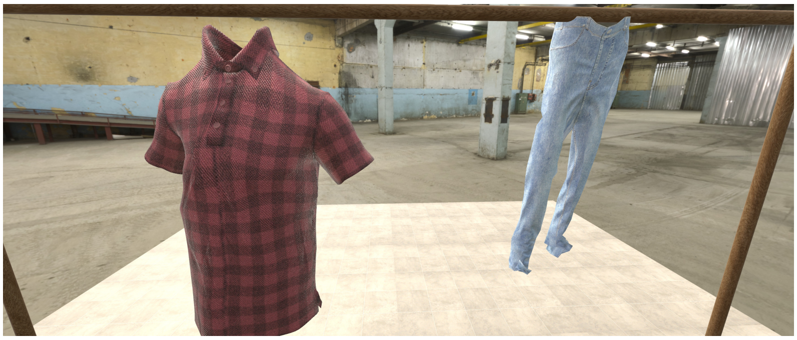
8. The implementation of the simulation is developed in context of a software engineering course with the following students: Niels Dossche, Glenn Feys, Jonas Bovyn, Tibo Vande Moortele, Stefan Croes, Julien Marbaix, and Laurens Debackere.



(a) A 100×100 rectangular cloth attached with two points to the bar. An invisible sphere on the right of the picture demonstrates sphere collisions.



(b) A 100×100 rectangular cloth draped on a table with spheres of different sizes.



(c) A simulated shirt and trousers from arbitrary meshes.

Figure 3.6 Spring-mass particle system for cloth demonstration.

3.4 Learning to fold in simulation

3.4.1 Deep reinforcement learning setup for cloth folding in simulation

In order to learn to fold cloth in simulation with a dual robotic arm, we use the DQN algorithm with small stabilization and accelerations improvements. We first describe DQN, followed by our task setup.

Deep Q-learning algorithm

DQN is a variant of the Q-learning algorithm (Watkins and Dayan 1992) using deep neural networks using a replay buffer and stabilization tricks, as described in Section 2.5. On top of the two stabilization improvements, we implement two additional ways to accelerate learning: prioritized replay and double DQN. Prioritized replay (Schaul et al. 2015) recognizes that not all observations in the replay buffer are equally important. By sampling the replay buffer proportional to the TD-error (see Equation (2.12)), experiences in which the agent made large prediction errors are more sampled. Prioritized sampling introduces bias that can be solved by annealed importance sampling. Double DQN (Van Hasselt, Guez, and Silver 2016) addresses overestimation bias caused by the argmax operator in the DQN update rule Equation (2.12). This is done by selecting the next best action using the online network while use the target network for getting the corresponding Q-value. The full pseudocode is given in Algorithm 1.

Task description

We define the task as performing a double fold in a rectangular piece of cloth by using a task structure. The tasks consist of grasping the corners of the cloth, then folding it inwards and releasing it. Afterwards, a second fold is executed by regrasping the corners and performing the final fold. The state space $\mathcal{S} \in \mathbb{R}^{114}$ consists of the 14 joint angles of the Baxter robot and

Algorithm 1: Double DQN with prioritized experience replay

```
Input: Discount factor $\gamma$, minibatch size $k$, target network
       update frequency $C$, prioritization factor $\alpha$ and
       importance sampling exponent $\beta$, learning rate $\eta$
1  Initialize prioritized replay memory $\mathscr{H} = \emptyset$, $\Delta = 0$, $p_1 = 1$
2  Initialize online action-value function $Q_\theta$ with random
     weights $\theta$
3  Initialize target action-action function $Q_{\theta'}$ with weights $\theta'$
4  foreach episode do
5      Observe $s_0$
6      foreach step of episode do
7          Sample action $a_t \sim \pi_\theta(s_t)$
8          Execute action $a_t$
9          Observe reward $r_t$, new state $s_{t+1}$
10         Store transition $\langle s_t, a_t, r_t, s_{t+1} \rangle$ in $\mathscr{H}$ with maximal
             priority $p_t = \max_{i<t} p_i$
11         for $j = 1$ to $k$ do
12             Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
13             Compute importance-sampling weight
                 $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
14             Compute TD-error $\delta_j = r_j +$
                 $\gamma Q_{\theta'}\left(s_j, \operatorname{argmax}_a Q_\theta\left(s_j, a\right)\right) - Q_\theta\left(s_{j-1}, a_{j-1}\right)$
15             Update transition priority $p_j \leftarrow \left|\delta_j\right|$
16             Accumulate weight-change
                 $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q_\theta\left(s_{j-1}, a_{j-1}\right)$
17         end for
18         Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$
19         Reset gradients $\Delta = 0$
20         Every $C$ steps: update target network $\theta' \leftarrow \theta$
21     end foreach
22 end foreach
```

the 10×10 positions of the cloth particles relative to the robot end-effectors. The action space $\mathscr{A} \in \mathbb{R}^{42}$ is in joint space: for each of the seven joints, the agent can move according to a fixed angle in any six directions. We provide a distance-based reward function $\boldsymbol{R}$ which incorporates the distance between the corner points of their current location and target location. Additionally, we include auxiliary objectives to provide a more rich reward signal to the agent: maximize the stretch of the cloth in order to minimize wrinkles.

$$\begin{aligned}\text{Cost} = & w_1 \sum_{p \in \mathscr{C}_f} \text{dist}\left(\mathbf{x}_p, \mathbf{x}_p^*\right) + \\ & w_2 \sum_{p \in \mathscr{C}_f} \text{dist}\left(\mathbf{x}_{\text{ee}}, \mathbf{x}_p\right) + \\ & w_3 \sum_{p \in \mathscr{C}_b} \sum_{p' \in \mathscr{C}_b, p' \neq p} \text{dist}\left(\mathbf{x}_p, \mathbf{x}_{p'}\right) \end{aligned} \tag{3.7}$$

The coefficients w_1, w_2, w_3 are weights for trading-off the reward function components. The particles in $\mathscr{C}_f$ are the particles for which we manually specified a target location in order to fold. $\mathbf{x}_p^*$ is the target location of the particle and $\mathbf{x}_{\text{ee}}$ is the current position of the end-effector. $\mathscr{C}_b$ are the particles lying at the boundary of the cloth. We aim to maximize the distance between the boundary nodes in order to minimize wrinkles. We multiply the cost function Cost with minus one for passing a reward for the agent. In our experiments, we represent the Q-function with a feedforward neural network containing two layers with 128 × 64 hidden neurons. We train a separate network for each of the four tasks. At test time, we evaluate the task sequentially and switch the network by detecting whether a subtask is solved. We consider a subtask to be solved when target particles are within 2.5 cm of their respective goals.

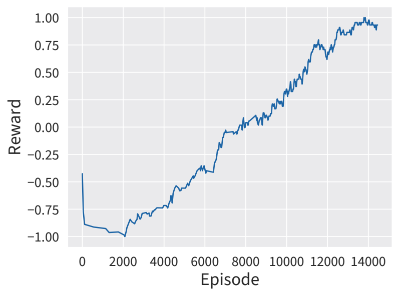
3.4.2 Results

When training the agent to learn how to fold a cloth twice, we notice a significant difference in learning speed between the different stages of the task. We are able to quickly learn how to grasp the cloth when it is in two unfolded configurations. In Figure 3.7a, the cloth initial layout is completely unfolded and the agent progressively learns to grasp the cloth. In the succeeding grasping task in Figure 3.7c, the cloth is already folded once. In this scenario, we see the agent learning the task at a similar pace to the first grasp. On the other hand, learning to fold when grasping the cloth is more difficult and requires roughly 31 times more interactions with environment as visible in Figure 3.7b and Figure 3.7d. The main reason for the folding task requiring more interactions compared to grasping can be found by considering the dynamics complexity of both tasks. The grasping tasks are essentially reaching tasks in which the agent must find the optimal path from the starting pose of the end-effector to the target particles. Once the agent finds the target locations, the particles are snapped to the end-effector due to undermodelling the contact dynamics. In contrast, during folding, the agent must find a trajectory that achieves the fold without causing wrinkles or sliding the cloth from the table. Small perturbations and random action exploration then leads to large changes in the received reward which renders the optimization landscape more difficult. Nevertheless, we find that the agent is able to learn the task within 24 h of training time. The success rates and training times we find are comparable to similar experiments in other work (Matas, James, and Davison 2018; Jangir, Alenyà, and Torras 2020). However, exact comparisons and benchmarking is difficult due to each work using different robotics simulators, cloth physics simulation models and learning algorithms.

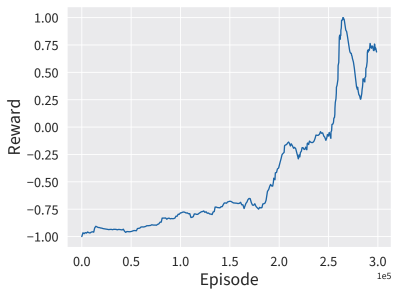
In Figure 3.8, we evaluate the success rate per stage in the task. We run this evaluation by letting the agent sequentially execute the task: grasping the corners, performing a first fold, regrasping the corners that are now at a different position and performing the second fold. This task is performed 100 times with different starting locations of the robot grippers. We find that the robot successfully executes this task with 97 % up to performing the second fold. During the last stage, the success rate drops due to the second folding operation failing 26 % of the time. We hypothesize this is due to the cloth being in a more difficult configuration containing more energy caused by the first fold. In addition, slightly hitting the cloth can cancel the resulting first fold whereas hitting the cloth in the first fold causes some local wrinkles. Finally, the first fold is easier due to resetting the cloth repeatedly to the same starting configuration. Contrarily, the second fold starts from the result of the first fold which contains more variability. We show visualizations of the learned grasping and folding behaviour in Figure 3.9.

3.5 Conclusion

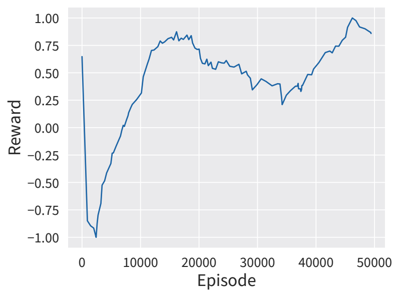
Simulation is an indispensable tool for the robotics researcher as it allows fast experimentation and massive, inexpensive data generation. The robotics simulator landscape is large, and choosing a specific technology requires specifying and trading-off domain-specific requirements. For our application of cloth folding, we found the support for cloth simulation in existing robotics simulators to be lacking. Therefore, we decided to use Unity as robotics simulator and implement a custom cloth simulation on GPU using a mass-spring methodology. We found that a DQN agent can learn to fold a rectangular piece of cloth twice in simulation within 24 h of wall time on standard computational hardware. However, it is unclear whether such an approach transfers to the real world. First, Sim2Real methods like domain randomization can be used for transfer but require a lot of computational power. Second, we require a lot of interactions in simulation. Learning from scratch on a real robot would take an exorbitant amount of time. Assuming a roll-out would consume one minute of robot time, this experiment would almost last half a year to complete. Third, the cloth state is used both as input observation for the agent and calculating the reward function. However, obtaining the state of the cloth in the real world is non-trivial. In the next chapter, we will provide a solution for these issues: estimating the cloth state using sensors and learning directly on the physical platform.



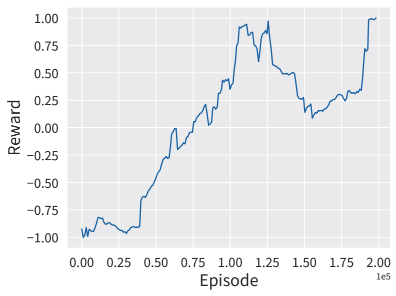
(a) Reward per episode for executing the first grasp.



(b) Reward per episode for executing the first fold.



(c) Reward per episode for executing the second grasp.



(d) Reward per episode for executing the second fold.

Figure 3.7 Mean reward per episode for the subtasks of learning how to fold a cloth twice in simulation. Each plot represents the progress for a certain subtask.

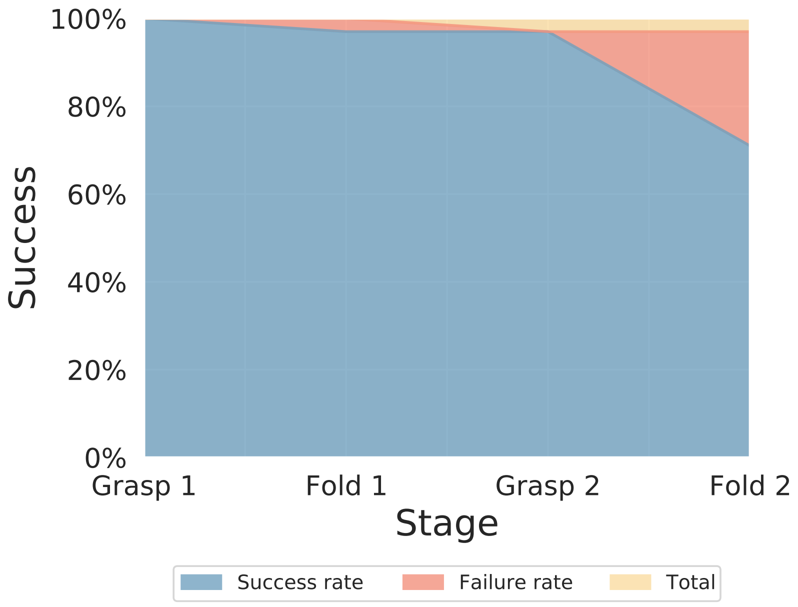
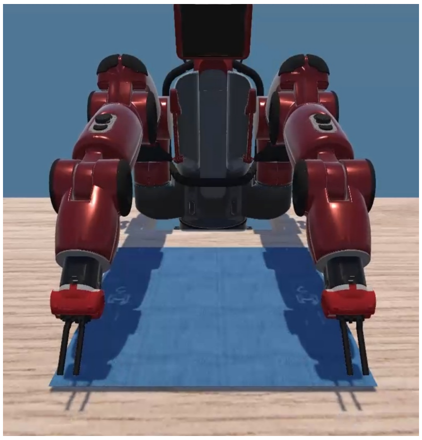
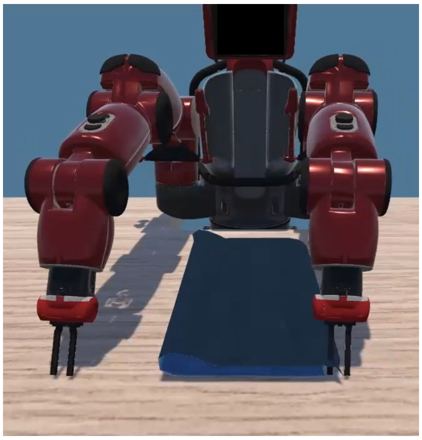


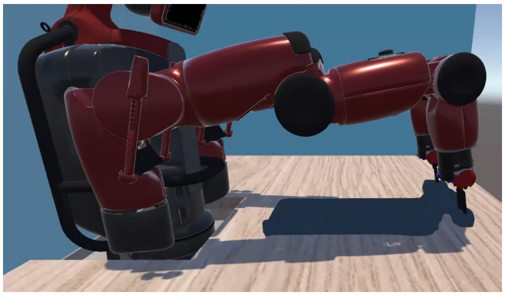
Figure 3.8 Success rate per subtask at test time for folding cloth twice in simulation.



(a) Baxter reaching towards cloth corner points.



(b) Baxter successfully performing the first fold.



(c) Baxter performing the second fold.

Figure 3.9 Visualizations of the learned grasping and folding behaviour.

4

4

Learning to fold through cloth instrumentation

We bridge the gap between simulation and the real world by estimating the state of cloth using smart textile technology.

4

Learning to fold through cloth instrumentation

In simulation environments, fully estimating the state of cloth is trivial because any point can be queried for information (Chapter 3). For example, bringing a cloth into a known configuration by grasping the corners can be done by querying the location of the corner points in simulation and planning a grasping trajectory. In the real world, finding corner points is more difficult due to sensor noise and self-occlusions. Chapter 2 concluded that existing folding research estimates the cloth state only in simulation or by using cameras and visual trackers in the real world. This chapter steps away from the vast focus on vision-based solutions and explores an alternative approach by embedding tactile sensing within a cloth and training a classifier to enable a smart cloth to communicate its own state. We show how this smart textile can be used for learning to fold a cloth on a low-cost, dual robotic arm.

First, we give a summary of our motivation for researching alternatives to vision-based cloth state estimation in Section 4.1. We refer to Chapter 2 in this book for a full review on the existing literature on cloth state estimation. Next, we introduce the robotic platform we utilize in our experiments in Section 4.2. Section 4.3 introduces our smart cloth technology and Section 4.4 discusses the real-world folding results we achieved with it. We provide a discussion on our future perspectives on grippers and cloth instrumentation in Section 4.5. Finally, Section 4.6 summarizes the main findings of our work on instrumentation.

4.1 Vision-based state estimation of cloth

When calculating the required trajectories for deformable object manipulation, the deformations need to be taken into account (Foresti and Pellegrino 2004). One approach to incorporate deformations is in simulation where the state is fully accessible. For example, Matas, James, and Davison (2018) use deep RL to train a neural network to learn to manipulate a towel. The reward signal consists of the simulated cloth state and the policies learned in simulation are transferred to the real platform using domain randomization. In their approach, the agent learns from the pixel input only, while tactile sensing coud lead to a more informational representation of the cloth. Another approach is to learn a forward dynamics model of the cloth in simulation (Tanaka, Arnold, and Yamazaki 2018). The learned dynamics model can then be used to bring cloth into the desired configuration. In our own work (Chapter 3), we calculated the distance between corner points that need to be close together in order to achieve a fold (Equation (3.7)). However, transferring results in simulation to the real world have proven to be challenging due to the difficulty in generating high-fidelity, simulated rollouts.

Sim2Real issues render state estimation of cloth in the physical setup a viable alternative. State estimation methods rely heavily on detecting points of interest like the corners of clothes. Maitin-Shepard et al. (2010), for example, regrasp a cloth until a corner point is found. However, this process is the largest bottleneck in their system, taking up to 24 minutes to fold a shirt. State estimation is also a requirement for RL approaches to cloth folding: a reward function is required to signal task success (Tsurumine et al. 2019; Matas, James, and Davison 2018). Balaguer and Carpin (2011) apply visual marks on a towel in order to track the state. The markings allow calculating the distance between points on the towel from the training sample and the example demonstrations so that it can be used as a reward for the agent. Their method requires prior information about the shape of the object in order to reconstruct the missing market points. However, relying solely on visual inputs and marker clues does not scale well. Another approach to find a reward function was explored in (Abbeel and Ng 2004; Finn, Levine, and Abbeel 2016), where inverse reinforcement learning (IRL) was used to learn a reward function from expert demonstrations. Because IRL learns a reward function and a policy separately, it requires RL to be run in an inner loop, making it computationally expensive to train. Learning the reward function offline decouples reward learning and policy learning and makes training on a real robot feasible.

To conclude, a typical approach for the robotic folding of textile relies on the use of vision in order to detect grasping points and to perform texture segmentation and pose estimation (Maitin-Shepard et al. 2010; Doumanoglou et al. 2016; Bersch, Pitzer, and Kammel 2011), as well as to estimate the state (Matas, James, and Davison 2018) and—in the case of reinforcement learning (RL)-based approaches—the reward function (Tsurumine et al. 2019). However, while vision is instrumental for recognizing and localizing objects, touch and force measurements become important once contact occurs and the object is explored using the end-effector (Billard and Kragic 2019). Recent work (Tian et al. 2019) has illustrated this idea by applying a touch-based control method for a ball repositioning task, rolling a dice and the deflection of a stick. In line with other authors (Tian et al. 2019; M. A. Lee et al. 2019), we believe the use of tactile input and its fusion with other sensory information is crucial to learn complex robot manipulation tasks in an efficient way.

4.2 Dual-arm robotic setup

Cloth manipulation is inherently bimanual because a second grasp is required for most tasks to bring cloth into desired configurations. There exist bimanual robotic platforms like Baxter® and PR2®. However, the spirit behind our research for instrumentation is one of democratization of hardware. A PR2 robot, for example, can cost $400.000. Sensors similar to the one we are proposing to construct, require advanced and financially expensive tools like photonic sintering systems (You et al. 2016). We aim to use off-the-shelf components and a DIY approach to instrumentation and robotics research. For this reason, we constructed the BCN3D Moveo robot arms[1]. The Moveo arm is an open-source initiative to enable the robotics community to build a low-cost robotic arm. The arm has five degrees of freedom. It can be fabricated by rapid prototyping techniques and off-the-shelf components. We show a picture of the constructed Moveo arm in Figure 4.1.

The task we propose is folding a rectangular patch of smart textile. We only consider the manipulation task of folding the cloth and assume the textile is already grasped by attaching the cloth to the two-fingered gripper by means of a hook-and-loop fastener. We place the arms opposable as visible in the setup in Figure 4.2.

4.3 Smart cloth

To detect the folded state of the cloth, we propose creating a smart cloth with integrated tactile sensing. We can then record a dataset of cloth states associated with sensor values to train a classifier that is able to recognize whether the cloth is folded. We first discuss the hardware options and our tactile sensing implementation, followed by our methodology and results to train a smart cloth.

4.3.1 Tactile sensing technologies

While tactile sensing always seems to be lagging behind compared to vision (Siciliano, Khatib, and Kröger 2008), the field has been growing in recent years (Chi et al. 2018). This growth can be attributed to novel fabrication methods (L. Zou et al. 2017) and a renewed interest in multimodal sensing for robotics (Lambeta et al. 2020). A complete overview of tactile sensing technologies can be found in (Siciliano, Khatib, and Kröger 2008) with updates about recent developments in (L. Zou et al. 2017; Chi et al. 2018). This section gives a short overview and discusses tradeoffs concerning various tactile sensing solutions.

1. https://github.com/BCN3D/BCN3D-Moveo

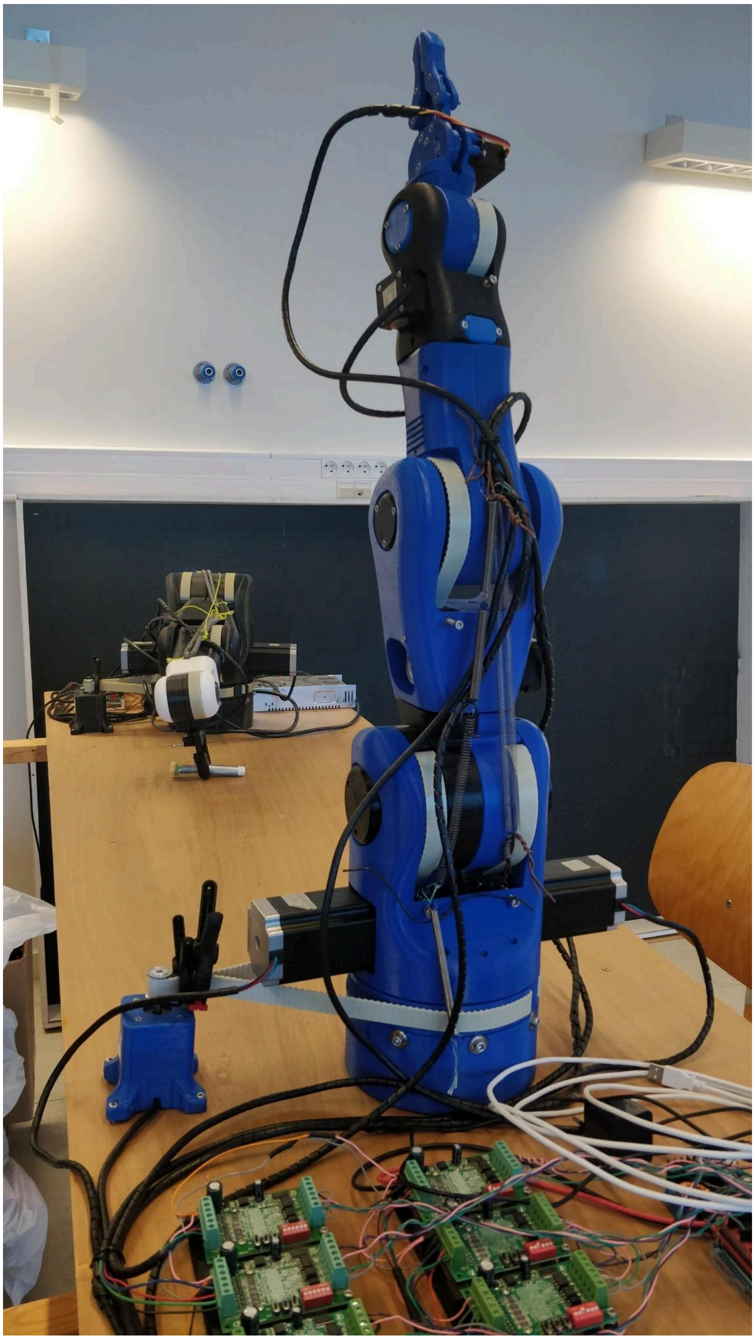


Figure 4.1 Picture of the constructed Moveo arm.

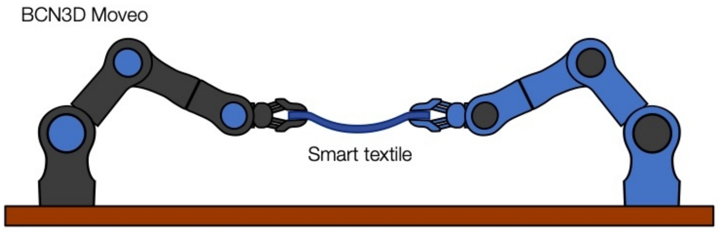


Figure 4.2 Schematic overview of the setup: dual robot arms hold a smart textile piece and have to learn to fold it from scratch using sensory feedback of the textile and proprioceptive input of the robotic arms.

The transduction mechanisms for external forces differ widely in the following properties: sensitivity, resolution, dynamic range, cost, ease of production and reliability. Common transduction mechanisms are capacitive, piezoresistive, piezoelectric and optical (Chi et al. 2018). Capacitive pressure sensing arrays form capacitors by separating a grid of electrodes by a dielectric material. A deformation in the dielectric material then induces a change in capacitance. Capacitive sensors are characterized by having high sensitivity and resolution but requires dealing with noise due to stray capacitances. Piezoresistive sensors operate on the principle of variable resistors: the change in resistance of a material acts as a proxy for force sensing. These sensors have a simple working principle and are low-cost to fabricate. For example, the strain gauge is a popular piezoresistive sensing technology available in many DIY electronics shops. Piezoelectric materials generate charges when being subjected to forces. However, the induced charges dissipate quickly, making it hard to use piezoelectric sensors to detect static contact. This is necessary when it is required to, for example, detect whether the cloth is still being grasped. Optical sensors analyze changes of internal light or track optical markers inscribed on a membrane. For tactile sensing with robot fingers, a growing class of visuotactile sensors measures the deformation of a gel or a membrane. Examples are GelSight (Yuan, Dong, and Adelson 2017), Soft-bubble (Kuppuswamy, Castro, et al. 2020) and Digit (Lambeta et al. 2020). However, we scope our requirements to tactile sensing to be embedded inside clothing. Hence, the used technology should exert minimal influence on the deformable properties of the textile.

For robotics in general, capacitive and piezoresistive are popular choices due to the mentioned advantages. Our application requires the sensing technology to be low-cost, accessible to fabricate and contain a high degree of sensitivity to detect folds. Piezoresistive sensors are simple and low-cost to produce, and have a high spatial resolution. The dynamic range can be configured with the fixed reference resistor in front of the variable resistor material. The low reproducibility of different piezoresistive sensors can be mitigated by using machine learning methods trained on different, uncalibrated batches of sensors. This makes piezoresistive sensing technology for our case more suitable compared to capacitive sensing, which requires relatively more complex measurements circuits to read out the capacitances and deal with noise and cross-talk.

4.3.2 Cloth sensing using piezoresistive rubber

The general principle behind the piezoresistive sensor is to construct a matrix of variable resistors. The voltage over the variable resistors is related to the pressure and can be calculated via the voltage divider principle shown in Figure 4.3. The voltage V over a variable resistor R becomes:

$$V = \frac{R_A}{R_A + R} \cdot \mathrm{V}_{in} \tag{4.1}$$

with R_A being the fixed resistor that can be optimized to maximize the range of V.

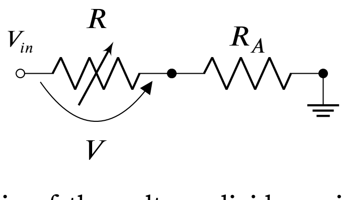


Figure 4.3 Schematic of the voltage divider principle. The voltage divider measures the voltage V over the variable resistor R and uses the fixed resistor R_A as reference.

Our implementation uses a piezoresistive rubber, commercially available as Velostat®. Velostat is a polymer injected with a carbonaceous powder called carbon black that provides electrical properties similar to graphite (Dzedzickis et al. 2020). By compressing Velostat, the carbon particles get pushed together, increasing the conductivity of the material.

The electrodes can be organized into a grid with a single or double layer. The first option to form a single layer of electrodes is visible in Figure 4.4. This design allows placing the electrical components on a flexible PCB with a Velostat layer on top. We show one of our implementations in Figure 4.5. The individual pressing points, called taxels, are measured by applying a reference voltage on the row electrode and iterating through the column points while measuring the input voltage on their input pins. The data is processed locally on the sensor by using an ATtiny1634 microprocessor. While this design integrates naturally for robotic fingertips, we found the electrodes in the copper wire to be fragile, the production cost is higher than two-layered copper threads and less scalable to integrate into the textile due to miniaturization on the flexible PCB.

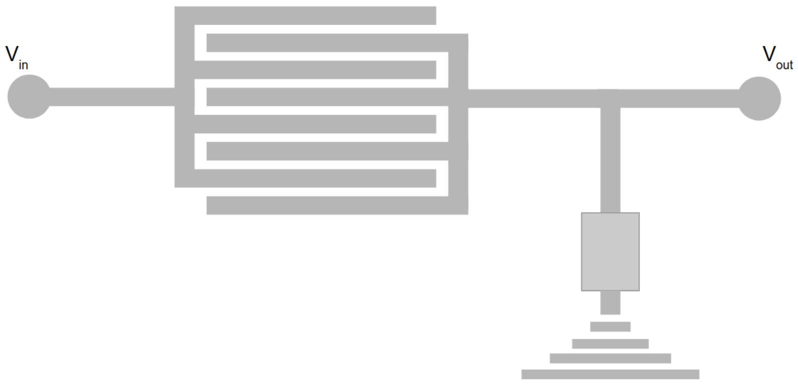


Figure 4.4 Schematic presentation of electrodes of a piezoresistive sensor organized in a single-layered pad.

Instead of organizing the electrodes in a single layer, we create a grid using a layer with horizontal electrodes and a second layer with vertical electrodes. The electrodes are made of copper strips that are insulated with a polyimide film. We sandwich the Velo-

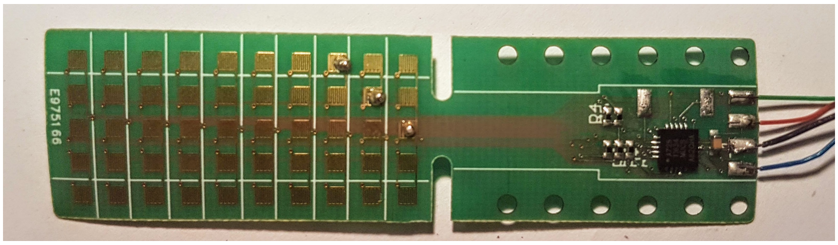


Figure 4.5 Our production of a piezoresistive sensor with electrodes in a single layer printed on a 0.2 mm thin, flexible PCB.

stat between these two conductive layers. The layers in the smart textile can be seen in Figure 4.6. This results in a matrix of 12 by 7 tactile cells. The signal conditioning for measuring the pressure applied over a tactile cell is based on the above-mentioned voltage divider principle. More specifically, the voltage is measured between the tactile cell, which acts as a variable resistor, and a fixed resistor R_A of 5.6 kΩ. A standard microcontroller, a Arduino MKR WiFi 1010, measures and processes the voltage. A LiPo battery powers the circuit and the wireless WiFi module of the Arduino allows communicating with the cloth remotely.

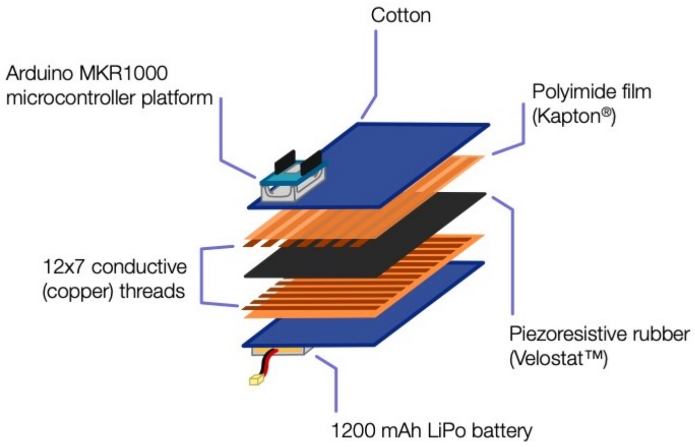

Cotton
Arduino MKR1000 microcontroller platform
Polyimide film (Kapton®)
12x7 conductive (copper) threads
Piezoresistive rubber (Velostat™)
1200 mAh LiPo battery

Figure 4.6 Exploded-view drawing of the textile with integrated tactile sensing. Conductive threads organized in a 12 by 7 grid are applied on piezoresistive rubber, creating a variable resistor. The resulting voltage acts as a proxy for local forces applied on the cloth.

Finally, the stacked layers of electronics is embedded in a cotton sleeve to form a smart cloth. The resulting textile can be seen in Figure 4.7. The electronics slightly influence the cotton's deformability, although we have not quantified the change in deformability. However, the primary deformable characteristic of the cotton is not changed, making it suitable for folding experiments. The main component making the cotton stiffer is the Velostat sheet of 0.1 mm thickness and weights roughly 15 g. The wires have a lesser impact because they run to the terminal endpoints of the copper electrodes, as can be seen in the zoom-in photo in Figure 4.8.

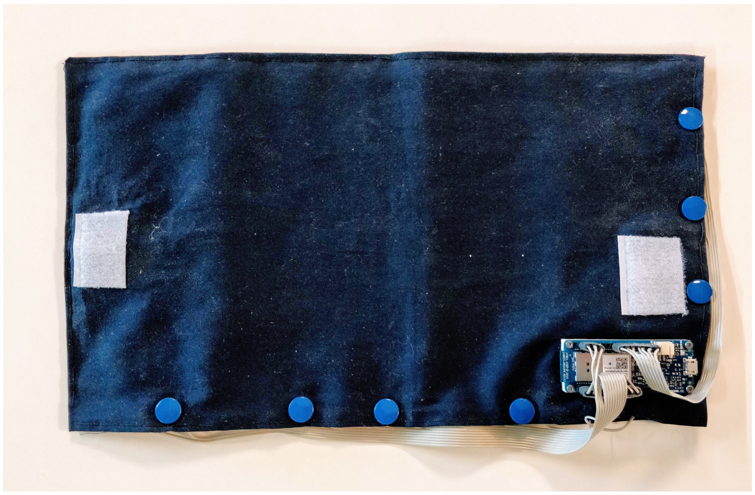


Figure 4.7 Our resulting implementation of a smart textile which is able to detect and translate local forces to a state configuration (for example, folded or unfolded). The smart textile is made using accessible off-the-shelf materials and tools.

4.3.3 Learning a smart cloth

We aim to use the integrated tactile sensors to make the textile able to sense and react to its surrounding environment, making it an active smart textile (Stoppa and Chiolerio 2014). To make the cloth smart, we train a linear classifier to predict the garment state: folded or unfolded. Therefore, we first collected data from the 12 by 7 tactile cells by bringing the cloth into multiple configurations while hand-labelling the associated state. We define

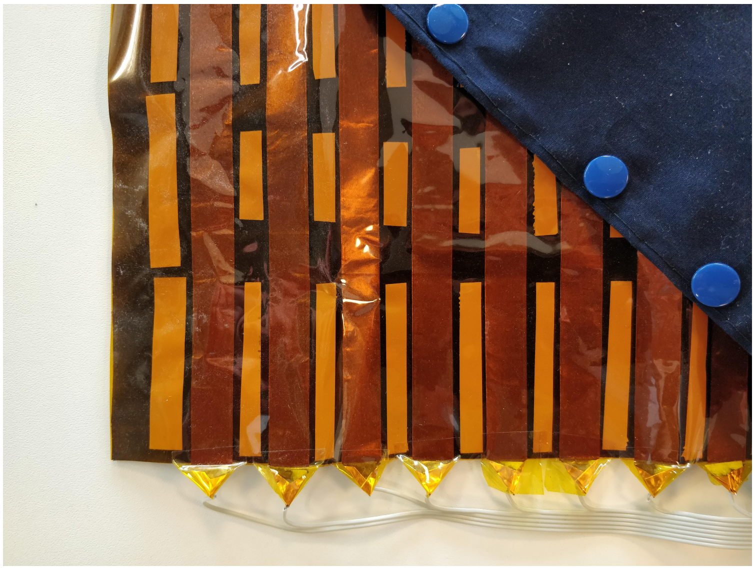


Figure 4.8 Zoom-in photo on the electronics embedded in our smart textile.

the folded state of the cloth with some slack because of the way in which the robots are positioned in order to avoid a collision; this is illustrated in Figure 4.9. The collected data were split into a training set of 1418 samples and a test set of 511 samples. In Figure 4.10, the average voltage and standard deviation of the tactile cells for both garment states are shown. One can observe that there is a clear voltage peak in the centre column, implying that the tactile cells in the centre of the textile experience higher pressure due to the folded state.

The classifier was trained using logistic regression with l2-norm regularization on 1418 training samples in a five-fold cross-validation setting. The regularization parameter was optimized by performing a random search and by using the F1-score as a performance metric. In order to verify the usability of the classifier, we tested it on an unseen test set of 511 samples which resulted in an average F1-score of 0.97. The full results of the test set are summarized in Table 4.1.

We are able to classify different shape configurations for the textile piece with high accuracy, making the smart textile useful

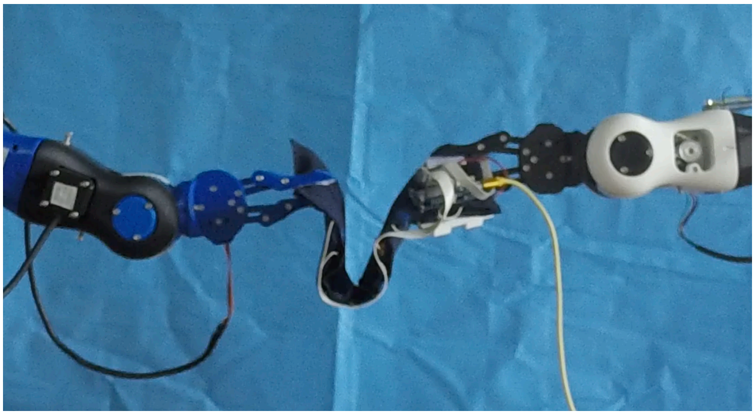


Figure 4.9 Example of the folding task on the real platform. The robotic arms are positioned in such a way that they cannot collide, making it physically impossible to fold the cloth fully. Thus, we do not require a fully folded state of the textile from the agent.

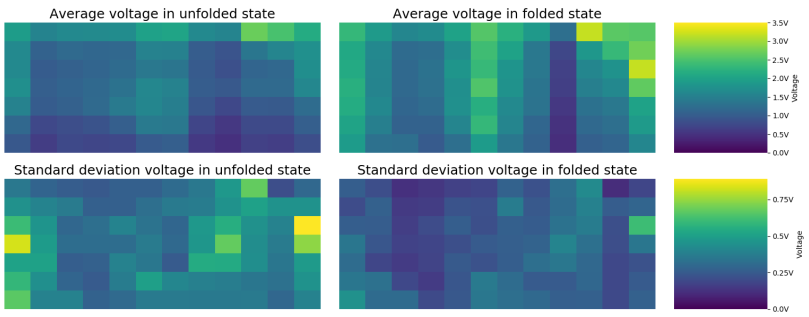


Figure 4.10 The average and standard deviation of the collected data from the 12 by 7 tactile cells show a higher voltage in the center column—where the fold is—when the cloth is in the folded state. These figures qualitatively indicate that the tactile cells in the textile react to different folding configurations.

Table 4.1 Test set results of the linear classifier that detects whether the cloth has been folded (511 samples).

	Samples	Precision	Recall	F1-score
Unfolded	342	0.98	0.98	0.98
Folded	169	0.96	0.95	0.96

for learning purposes. Other desired features such as wrinkle detection are hard to measure with this technology given the limited sensitivity of the piezoresistive rubber.

4.4 Results on learning to fold an instrumented cloth

As described in Section 4.2, the task consists of folding a rectangular textile piece by using a low-cost dual robotic arm. The arms are positioned in an opposing manner to avoid a collision. In order to avoid damaging the smart textile, we limited the movement of each robot arm to three degrees of freedom (DOF) instead of the available five DOF. Consequently, the movements of the robots are limited to one plane.

In order to find the control policy for folding, we formulated the problem as a fitted Q-learning problem (Watkins and Dayan 1992). The state-space $S = \mathbb{R}^6$ is defined by the joint space of the dual-arm robot, with six motor angles in total. For each joint, we define two actions: $\pm\Delta$ with Δ equal to 4°. The reward function R is sparse and defined by

$$R = \begin{cases} 100 & \text{on success} \\ -1 & \text{penalty for wandering} \end{cases} \tag{4.2}$$

We approximate the Q-function with a simple MLP with six input neurons (one for each of the controlled joints), two hidden layers with 128 and 64 ReLu neurons, respectively, and 12 output (linear) neurons (one for each of the possible actions). In order to find a good set of parameters, we implemented a simple forward kinematic model of the robot arms in a reaching task setting. This allowed us to sweep across various hyperparameter settings. The optimizer we used is stochastic gradient descent with a learning rate of 0.01 and batch size of 64 samples. The target network is updated each 1000 steps. Exploration is done using an ϵ-softmax policy with 5000 exploration steps. Thus, the agent will first execute fully random motor actions and gradually start choosing actions that maximize the expected q-values.

After the selection of the hyperparameters, we trained our dual robotic arm setup from scratch in the real world. As shown in Figure 4.11a, after approximately 60 episodes (17.000 steps, or 8 h), the robot successfully learned to complete the folding task. Figure 4.11b shows the number of steps the robot needs to fold the cloth per episode. The low dimensional state space of the agent and the learned reward function from tactile information using supervised learning allows the agent to learn to fold the cloth after relatively few training examples.

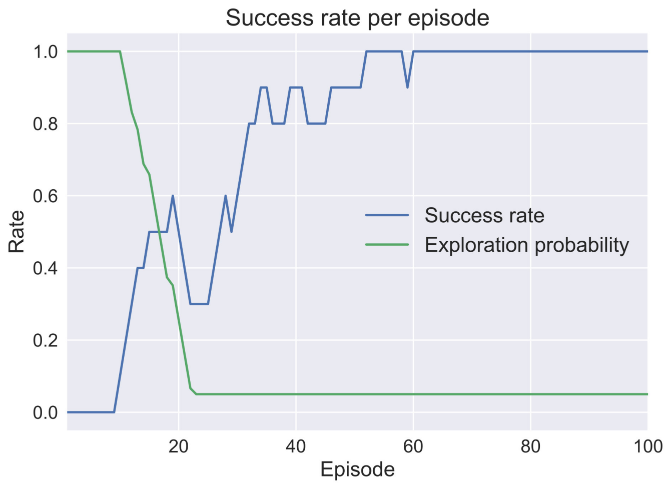
An example trajectory of the learned policy is shown in Figure 4.12. The action probabilities show that at the start of the episode, the agent primarily moves the shoulder joints, as they lead to the largest movement of the end-effector. This behaviour has the benefit that it minimizes the penalty associated with wandering or other suboptimal movements. As the episode progresses, the agent has a higher probability of actuating the elbow and end-effector motors for more refined movement. The end-effector trajectories, shown in yellow, are not smooth given that the policy is an action probability function calculated over the Q-values of the neural network. Because there are multiple reaching points in the plane to fold the cloth, the agent has multiple options to move during the first part of the episode, leading to jagged end-effector trajectories.

As our experiment illustrates, it is possible to avoid the need for complex vision-based state estimation to learn difficult tasks such as the folding of clothing with low-cost robots. The use of other input modalities such as a smart textile allows a reward function to be learned with simple linear models and is applicable to use for learning on real robotic platforms.

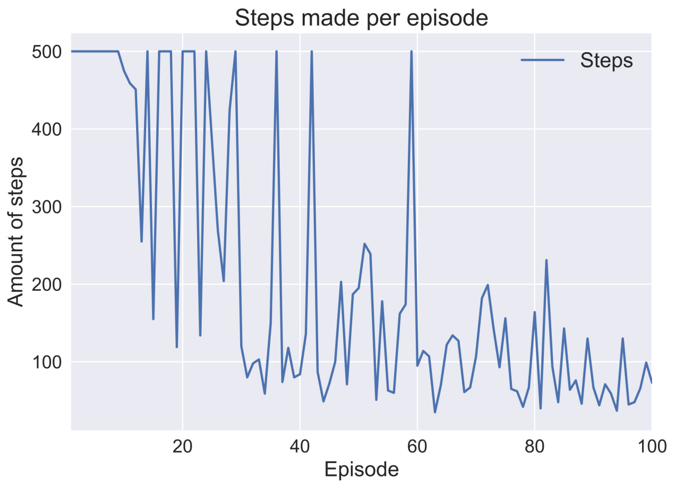
4.5 Discussion

4.5.1 Grippers for cloth manipulation

The task we solved in this setup was essentially one of low complexity to demonstrate the applications of accessible sensing technology. When considering more complex cloth-related tasks



(a) The folding success rate and exploration probability per episode. The success rate grows to 100% after training for 8 h on the real robot.



(b) The number of steps required per episode to solve the task; the number of motor commands needed to fold the cloth decreases after relatively few training examples.

Figure 4.11 Results of training the real robot from scratch to learn how to fold a smart textile.

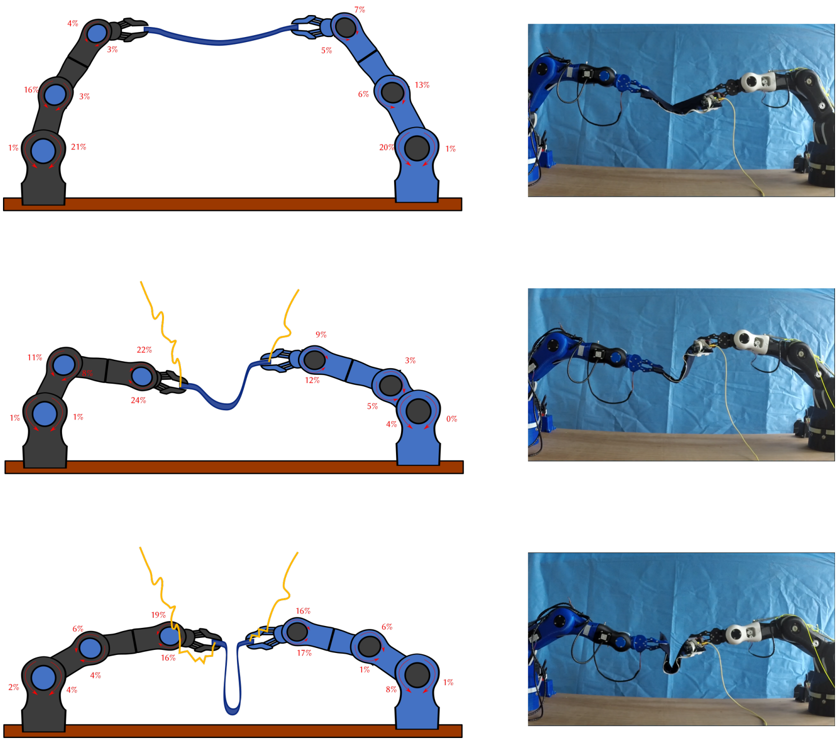


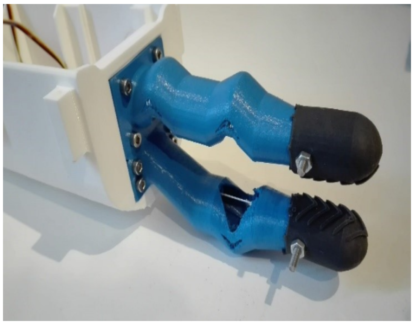
Figure 4.12 Demonstration of a folding trajectory of the learned policy broken down into three steps: start, middle and end of the episode. The robot state on the figure is visible from the stance of the robot arms. The action probabilities are given in red next to each joint. The executed trajectory is shown with a yellow line.

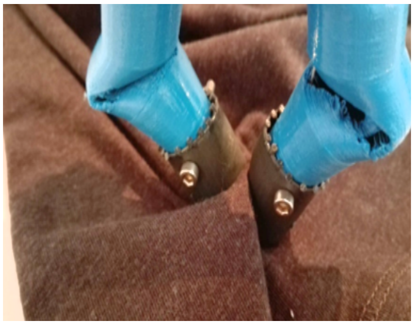
like grasping and flattening, the question of which gripper design is appropriate arises. We have experimented with different underactuated designs, illustrated in Figure 4.13. However, it is non-trivial to determine what types of end effectors and grasps are needed for optimally manipulating cloth-like objects. Recently, Borràs, Alenyà, and Torras (2020a) provided a holistic overview of the known gripper designs for manipulating textile. In this work, a systematic classification of grasps in the cloth literature is given based on opposing geometric primitives. Their work brings the important observation that most prior research uses a pinch grasp with both general-purpose and specialized cloth-specific grippers. The proposed framework is used to demonstrate that adapting the gripper type to the cloth manipulation task lead to more robust control. We believe this initiative to be a good starting point before resorting to arbitrary cloth manipulation tasks with off-the-shelf parallel grippers.

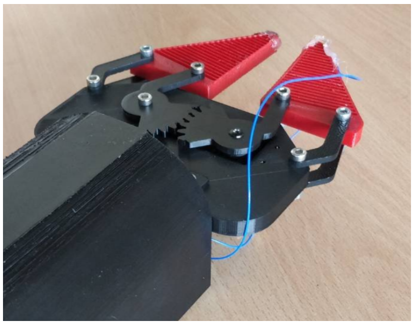
4.5.2 Future improvements

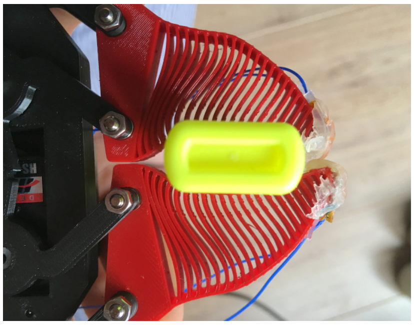
The instrumented learning process proposed in this chapter is feasible to scale up to other instances of cloth manipulation tasks given the low computational requirements and the tactile sensitivity of the sensor matrix. The accessible fabrication procedure of the smart textile allows the implementation of this technology for any type of clothing, such as shirts and trousers. The influence on the deformable properties of the cloth can be reduced by miniaturizing the electronics, by using conductive thread instead of copper wires, and by only applying tactile cells in crucial areas. The Arduino we mount on the cloth contains unused functionality and can be replaced by a smaller PCB customized for this task.

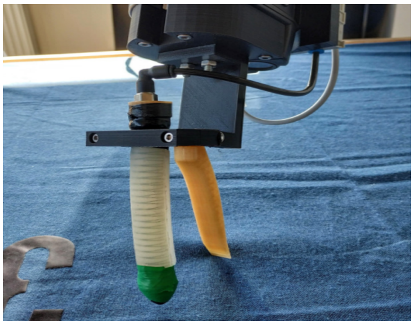
Future work includes exploiting the sensitivity of the tactile sensor grid to classify more complex cloth configurations and using more powerful, non-linear classifiers compared to the logistic regression model used in this work. Future research should be directed towards fusing the tactile information with visual data in order to generate a database of self-labelled images of arbitrary


(a) Underactuated finger printed with flexible PLA material.


(b) Finger tips with profile to increase friction for grasping.


(c) Gripper based on the deformation of the fin of a fish (Crooks et al. 2016).


(d) Compliance increases safety and adapts to the shape of the object.


(e) Pneumatic gripper.

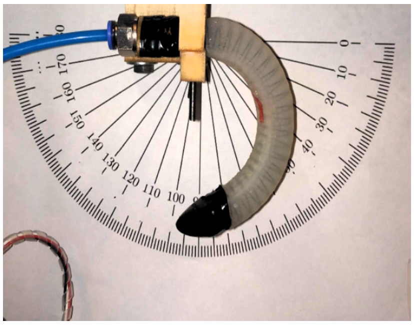

(f) Bending range.

Figure 4.13 Alternative gripper designs we experimented with for cloth manipulation tasks. In collaboration with Peter Verdru, Robbe Nuyttens and Stijn Fierens, former master students at our lab.

cloth configurations. These data can be produced by manipulating the cloth while recording its state with cameras. In order to detect and grasp the cloth and train policies that minimize wrinkles, visual feedback will have to be included. A suitable approach for integrating visual and tactile information is to learn a multimodal latent space, as shown in (M. A. Lee et al. 2019). We discuss our ideas for multimodal learning in Chapter 7.

4.6 Conclusion

The difficulty in estimating the state of clothes causes problems for robotic vision pipelines and hinders making reward functions for RL. In this chapter, we proposed a cost-effective and computationally efficient solution for estimating the state of cloth. We have shown that an inexpensive dual robotic platform can learn to fold a textile piece in the real world with neural fitted Q-learning and without reward engineering. This is done by including tactile sensing in the learning process. We have shown that the high sensitivity of the proposed tactile cells allows the capture of arbitrary cloth configurations. We demonstrated that the tactile data can be used to train a simple logistic regression classifier to detect the cloth state; thus, we create a smart textile piece that is able to accurately detect whether it is folded.

We are aware that we will have to integrate vision for the state estimation to scale up our tasks to real clothing. Consequently, we do not advocate vision-free solutions, but we argue for an interdisciplinary approach in which vision, proprioception and tactile information is combined in order to find the simplest solution of state and reward function estimation.

An important bottleneck of the research presented in this chapter can be found in the reward function in Equation (4.2). The scalar value received each step, and on completing the task, are arbitrarily chosen. Our experiments from Chapter 3 also confronted us that the issue of reward shaping (discussed in Chapter 2) is also present for cloth folding tasks. We believe that learning the reward function from demonstrations may overcome human bias in reward engineering. However, as discussed in Section 2.6, the prerequisite for learning is unavailable; there exists no high-quality dataset of people folding clothing. The lack of a folding dataset is tackled in the next chapter.

5

5

Dataset for robotic folding through crowdsourcing

We collect the first large-scale video dataset of people folding clothing to bootstrap learning in robotics.

5 Dataset for robotic folding through crowdsourcing

Deep neural networks lie at the heart of modern methods to acquire complex robotic manipulation skills. For example, Matas, James, and Davison (2018) use a deep CNN that maps input images to motor commands to fold cloth. However, a prerequisite for learning is the availability of data. In the case of using deep learning models, the amount of data required is considerably high. Unfortunately, datasets for robotic learning of manipulation skills for clothing and deformable objects are not widely available. While a large body of research exists on cloth modelling and garment reconstruction from images (Bertiche, Madadi, and Escalera 2020; H. Zhu et al. 2020; T. Y. Wang et al. 2018), the availability of dedicated cloth folding data is scarce. To fill in the gap of the lack of a high-quality and high-volume cloth folding dataset, we collect a dataset of people folding clothes. Our dataset fulfils two roles: (1) we aim to provide the community with a high-quality and heterogeneous dataset of people folding clothing; and (2) this dataset connects the former and next chapter in this book where we learn, instead of engineer, a reward function.

5.1 Crowd-sourcing folding demonstrations

We gathered a heterogeneous dataset of folding demonstrations using a community-based participatory approach (English, Richardson, and Garzón-Galvis 2018). We involve citizens by requesting them to demonstrate their method to fold clothing

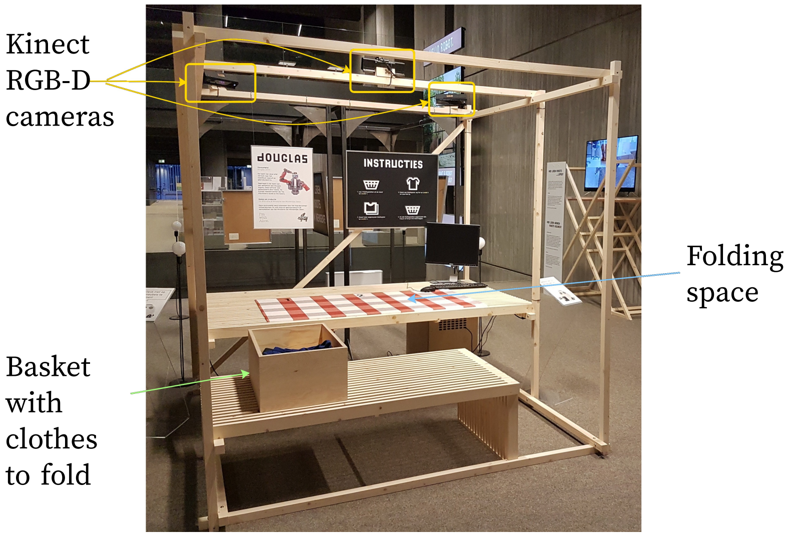


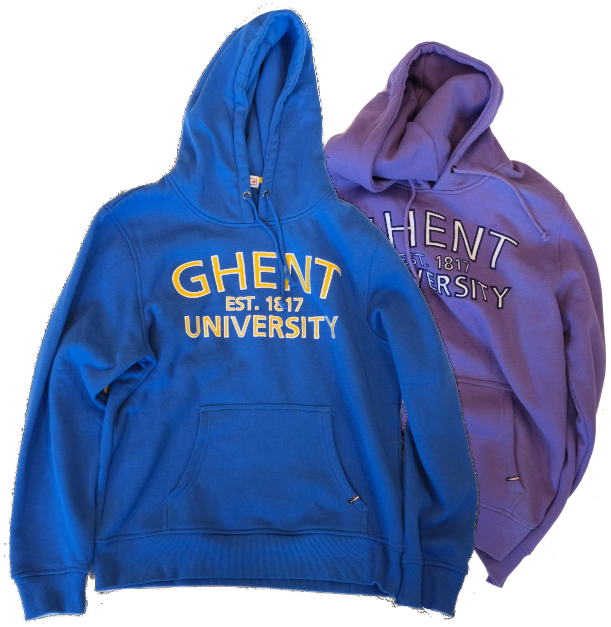
Figure 5.1 Picture of our folding table setup to crowdsource video demonstrations in a public library. There are three Kinect cameras mounted on top of the table that capture the human folding demonstrations. We provide easy and clear instruction to introduce consistency in the demonstrations.

on a folding table with cameras. Using posters, an instructional video and warning symbols around the folding table setup, we made it explicit that participants will be recorded on video for the purpose of research in the domain of robotics and AI. We collected no demographics or other personal information. This setup allows us to capture different folding strategies and manipulation varieties within a folding method. The participants consist of a combination of students and visitors of a public library in the third-largest city in Belgium. This avoids selection bias in the dataset. Furthermore, we place our setup within a small exhibition on research in robotics to inform the public about learning strategies for robots and give an answer to an innate fear in society that self-learning robots could lead to a loss in jobs (Fleming 2019).

To capture video task demonstrations, a special-purpose folding table was designed and constructed, which can be seen in Figure 5.1. The table is a beam-like, wooden skeleton structure consisting of a tabletop, a bench, camera mounting points, a basket,

(a) The towels in the dataset contain different textures and are of similar size.

(b) The sizes of the t-shirts range from small to extra large and consists of a multitude of colours.



(c) The hoodies are arguably the hardest piece of textile to fold in the set. There are two hoodies with a different colour.

Figure 5.2 The set of textiles that has to be folded by the participants consists of hoodies, t-shirts and towels with a variety of sizes, colours and textures.

and a locker. The participant is required to fold the clothing on the working surface. The tabletop is detachable in order to apply different tablecloths as a means to introduce additional variety in the dataset. As we require the demonstrator to sit while performing the task, we place a large bench attached to the wooden frame. The bench also obstructs observers to prevent occlusion and distraction during task execution. There are three Kinect v2 cameras mounted on top of the table. They capture the perspective from the task executor and two top corner video streams to deal with occlusion. They are placed approximately 160 cm and 183 cm from the centre of the folding table in order to capture the complete folding sequence demonstration. The Kinect cameras provide RGB and depth information at a resolution of respectively 1920×1080 and 512×424 pixels. The wooden basket is attached to the bench and serves as a proxy for a laundry basket. Finally, a locker safeguards the workstation embedded in the table. We use the libfreenect2 driver (Xiang et al. 2016) to capture the frames and process the six video streams, RGB and depth information, online using an AMD Ryzen 1700X CPU. Because of the high bandwidth requirements of the Kinect cameras, we limit the frame rate to 10 FPS.

To structure the participants' task demonstrations, we provided a four-step instruction list: (1) place randomly selected clothing out of the basket on the left side of the table, (2) fold one textile at a time in the middle of the table, (3) collect it at the right of the table and (4) put all textile back in the basket. We made an instructional video[1] and put up a poster containing these instructions to avoid high variance in task execution.

Because the folding table is stationed in a public space with mostly no human supervision, we leave it recording throughout the project. To avoid running out of storage and filtering frames without human activity, we run an activity detection heuristic based on changes in the pixel values of the video stream. To guarantee ample observations, we also actively visited the setup to attract and inform visitors about the project. We noticed our presence had a positive effect on the number of visitors willing

1. The instructional video can be viewed at https://www.youtube.com/watch?v=p99T5H8yK38.

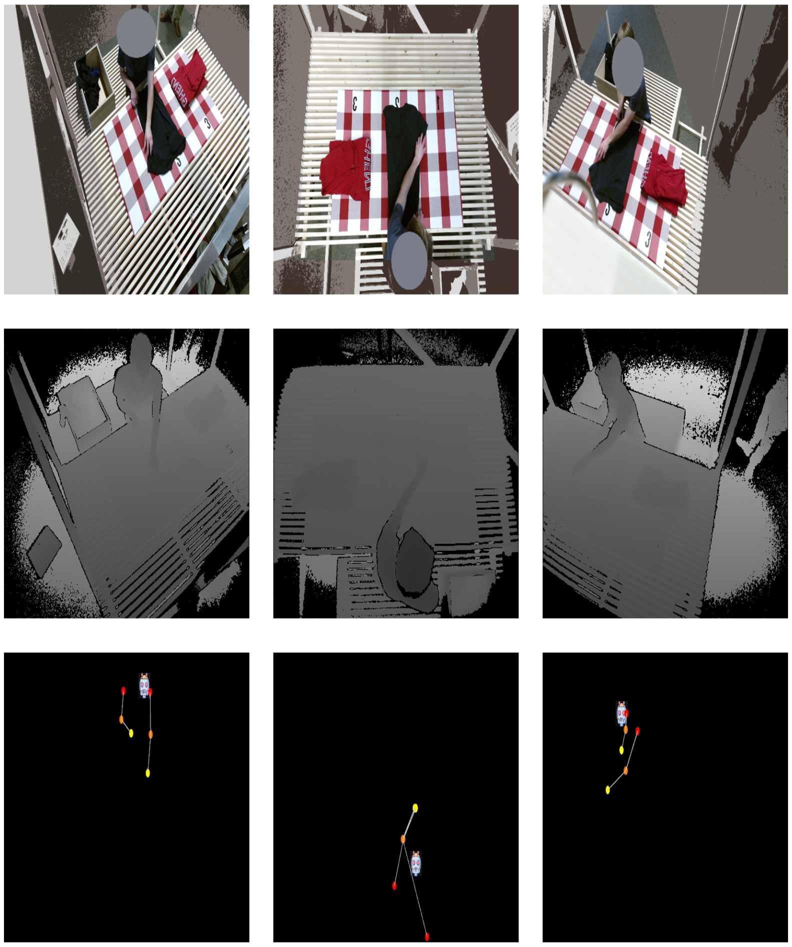


Figure 5.3 **Example output from our dataset.** We provide RGB images, depth registrations and skeleton keypoint trajectories of 6.5 hours of human demonstrations of folding clothing. The RGB images are anonymized without compromising image fidelity or disturbing the folding demonstration. The videos are segmented such that one sample represents the folding of one piece of textile.

to participate in the data crowdsourcing project.

The included types of textiles in the basket are towels, t-shirts, and hoodies. Examples are shown in Figure 5.2. We excluded trousers as they are hard to flatten from a sitting position. Socks were also excluded from the set because folding socks require high-dexterous, fine manipulation actions that would not be visible from the mounting position of the cameras.

After capturing the example demonstrations, we cleaned all false positive recordings from the database, sliced the recordings into single-piece folding demonstrations and manually labelled subtasks. The subtasks consist of grasping isolated clothing, unfolding, flattening, folding and stacking it on top of each other. We defined exact definitions of these subtasks in Table 5.1 in order to consistently label the video fragments. As data quality plays an important role for learning algorithms, we annotated the data ourselves to ensure there is consistency in the labelling between samples. These subtask labels can, for example, be used in reinforcement learning for reward engineering or hierarchical learning and for the training of action recognition systems. Skeleton keypoints were extracted from all frames in post-processing using AlphaPose (Fang et al. 2017).

5.2 Folding demonstrations dataset

The observations in the datasets are captured over the course of two months at two different public locations. The set contains 1000 folding demonstrations of three different types of textiles. This amounts up to 8.5 hours of folding recorded in 304.820 frames. We registered four different types of folding methods. We segment each video into chunks of single folding demonstrations and provide RGB frames, depth information, annotations, pose trajectories and timestamps of the different steps in the folding task. The content and how to access and use our dataset is described in the remainder of this section.

Subtask	Definition
Isolated grasping	The subject selects grasping points to remove a piece of textile from a heap of multiple textile pieces and isolates the selected textile.
Unfolding	The subject selects grasping points and executes manipulation trajectories in order to remove a fold.
Flattening	The subject executes manipulations in order to remove wrinkles from a piece of textile which can be in any state.
Folding	The subject selects grasping points and executes manipulation trajectories in order to bring the textile into a folded configuration.
Stacking	The subject grasps the textile and moves it outside the folding area, possibly stacking it on top of a pile of folded textiles.

Table 5.1 Definitions used to label the subtasks in the folding task

5.2.1 Folder structure

The dataset is segmented into folding demonstrations of a single piece of textile. This structure is visible in the folder hierarchy in Figure 5.4. For each example demonstration, we find annotations and timestamps indicating the subset of the task. Because we have three cameras mounted on fixed positions on the folding table, we put the colour, depth and pose registrations in the folders *left*, *middle* and *right* which represent the viewpoint in front of the table.

5.2.2 Data format

The RGB images captured with the Kinect cameras are compressed with the *x265* codec. We use the intrinsic camera calibration parameters to modify the images according to the depth correction. All RGB frames are anonymized by applying colour quantization to the corners of the frame and pasting an

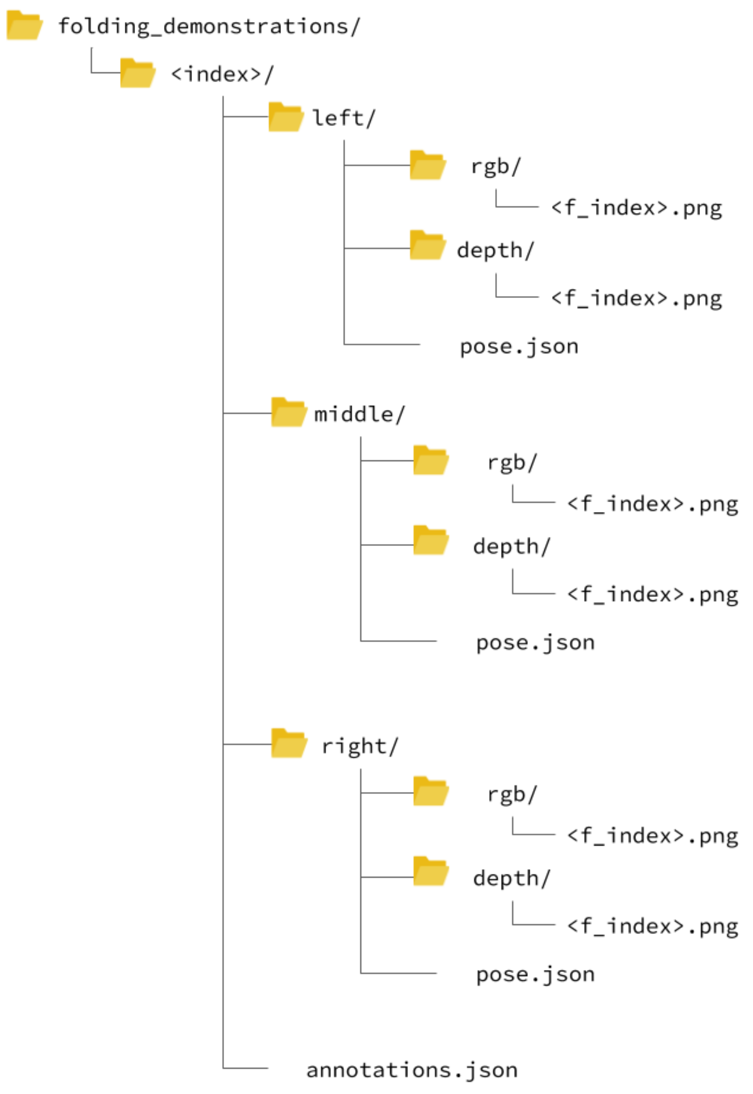


Figure 5.4 Folder structure of the folding demonstration dataset. There is a folder per folding demonstration, indicated with <*index*>. Each sample contains labelled data in the annotations json file. The images are grouped per perspective and contain rgb and depth images. There are also joint positions available per video perspective.

ellipsoid colour patch around the face of the demonstrator which was tracked using AlphaPose (Fang et al. 2017).

Each sample in the dataset contains annotations in the *annotations.json* file. The labelled information and data format can be seen in Listing 1. We label which type of textile is being folded and which folding method is being used. We distinguish four categories of folding methods, labelled from *a* to *d*. These categories represent an increasing amount of complexity to learn a certain folding strategy. For example, folding method *a* extensively uses the table to make folds. In contrast, method *b* represents demonstrators making vertical folds while lifting the cloth in the air. Method *c* categorizes folding strategies which requires crossing the hands. Finally, method *d* captures all different strategies not described by the former folding categories, for example rolling up the cloth. All four folding strategies can be used on all types of cloth in the dataset. The different types of textile are labelled as *hoodie*, *shirt* or *towel*. The distribution of the folded clothing is as follows: 88% of the folded clothing are shirts, 9% are towels and 3% hoodies. The timestamp is in `YYYY-MM-DD HH:MM:SS` format. Given that the data is crowdsourced, some variation exist in the way participants followed the high-level process instructions. For example, some demonstrators fold the clothing immediately out of the basket instead of first collecting the pieces on the left side of the table. To indicate to which extent the given process instructions are being followed, we included a quality label in the annotations. This label is useful if consistent, high-quality samples need to be sampled. In the dataset, 86% follow the given instructions, 12% make one deviation while 2% do not follow the given high-level instructions. The exact definitions of the quality label are shown in Table 5.2.

We labelled each part of the video with a descriptor indicating which step in the folding process the demonstrator is going through. The different steps are named *isolated_grasping*, *unfolding*, *flattening*, *folding* and *stacking*.

We provide human skeleton keypoint trajectories in the file *pose.json*. There are pose trajectories available for each camera perspective per folding sample. The joint positions are stored in the JSON format visible in Listing 2. There is a score associated

Quality label	Definition
Follows instructions	The instructions were followed exactly.
Slight variation on instructions	One deviation was made from the instructions.
Very different from instructions	Two or more deviations were made from the instructions.

Table 5.2 Quality label definitions: because not all demonstrators follow the given high-level task instructions, we define a quality label of which the definitions are given in this table.

with each pair of x and y coordinates. This variable, ranging from 0 to 1 indicates the detection confidence that a certain joint is at the given location. In general, the coordinates for every joint positioned beneath the shoulders are less reliable because the subject is sitting on a bench with the legs occluded by the table. We consider this not a problem because the coordinates of the joints of the two arms are reliable and are of importance for the folding task.

5.2.3 Project website and helper scripts

Along with the data, we provide helper scripts in Python which are available at https://github.com/adverley/folding-demonstrations. The data can be loaded by calling `FoldingDemonstrationDataSet(home_dir)`. We expose the data as a nested dictionary, embedded in a list. This enables an intuitive interface for accessing the data by iterating over the `FoldingDemonstrationDataSet` object and querying specific fields with corresponding key in square brackets. For example, `data[0].annotations['clothing_type']` queries the type of clothing being folded in demonstration 0 while `dataset[42][0]['rgb']['left']` returns the first RGB image of video demonstration 42. A more complete and general-purpose example can be found in Listing 3.

```
{
    "id": 0,
    "timestamp": "2018-09-30 19:35:06",
    "cloth_type": "hoodie",
    "location_id": 0,
    "nb_frames": 579,
    "folding_method": "a",
    "demonstration_id": 0,
    "nb_folds": {
        "0": 0,
        "66": 1,
        "94": 2,
        "118": 3
    },
    "subtask_changes": {
        "0": "isolated_grasping",
        "42": "unfolding",
        "112": "folding",
        "300": "stacking"
    },
    "quality": "follows instructions"
}
```

Listing 1 annotations.json description that provides metadata about a single folding demonstration.

5.3 Conclusion

In this chapter, we introduced a video dataset with human demonstrations of folding textile, captured via a citizen crowdsourcing project. With this unique dataset, we aim to fill a gap in learning deformable objects manipulation, bootstrapped by human examples. We provide 1000 demonstrations with RGB images, depth frames, and joint pose trajectories captured from three perspectives simultaneously. We labelled the data with subtask annotations, folding method, and textile type. Our goal is to provide robotics researchers with a real-world

```
{
    frame_nr: {
        "LElbow": [x, y, score],
        "RElbow": [x, y, score],
        "LShoulder": [x, y, score],
        "RShoulder": [x, y, score],
        "LWrist": [x, y, score],
        "RWrist": [x, y, score],
        ...
        "confidence": 100.0
    }
}
```

Listing 2 pose.json description that gives image coordinates of the joints of demonstrator folding clothing.

dataset to accelerate the learning from human demonstrations for deformable object manipulation.

Our dataset can be employed for multiple purposes. A first use-case is action recognition on multiple temporal levels, for example detecting different folding methods and detecting the steps associated with folding textile. A second application is to use the data to bootstrap learning via learning from demonstration. We explore the latter in the next chapter.

```python
from folding_demonstrations.dataset import
 ↪  FoldingDemonstrationDataSet

# Set to the directory where the folding
 ↪  demonstrations dataset is stored
home_dir = '/media/data/folding_data_output'

# Load the data
dataset = FoldingDemonstrationDataSet(home_dir)

# Iterate over data and query available
 ↪  information
for sample in dataset:
    random_frame_nr = 42
    frame = sample[random_frame_nr]
    rgb_left = frame['left']['rgb']
    rgb_middle = frame['middle']['rgb']
    rgb_right = frame['right']['rgb']
    depth_l = frame['left']['depth']
    depth_m = frame['middle']['depth']
    depth_r = frame['right']['depth']
    subtask = frame['subtask']
    reward = frame['reward']
    pose = frame['pose']
```

Listing 3 Example code how to query the dataset

6

6

Learning reward functions from demonstrations

We present the first system that expresses the progress of people folding clothing without labelling video frames.

6 Learning reward functions from demonstrations

We employ our dataset of people folding clothes (Chapter 5) to distil the task intent and express it as a scalar value. This scalar value is a metric representing task progression which can be used for monitoring processes or as reward function in RL. To achieve this reward metric, we present a method that uses the multiperspective video demonstrations to train an embedding by using time as a contrastive signal. This self-supervised approach enables label-free learning. By aligning demonstrations to a small set of hand-picked expert samples, we can extract a reward signal. The contributions of the research presented in this chapter are threefold:

1. *A novel method to generate task progression metrics from video without labelling data:* We propose an integrated approach to overcome expensive data labelling in data-driven process monitoring in order to generate self-learned task progression metrics. Our approach also allows decoupling reward and policy learning in reinforcement learning.
2. *The first solution for tracking cloth folding progression*: We provide the first results for the challenging case of quantifying cloth folding progression.
3. *In-depth, case-based robustness analysis:* We demonstrate the robustness of our approach with adversarial cases to test the use-cases and limits of our proposed method.

This chapter is organized as follows. First, we describe our rationale and related work in Section 6.1. Next, we give a high-level overview of our proposed approach in Section 6.2 and then discuss all components in Section 6.3. We demonstrate quantitative results of learning progression metrics in Section 6.4. The discussion sections of this book frequently touch on the subject that a progression metric for folding cloth is ill-defined. For this reason, we provide an in-depth discussion in Section 6.5. This discussion decodes which visual features the progression metric is attending to in the scene. We also conduct case-based adversarial experiments to test which invariances are learned and how robust our method is. Finally, Section 6.6 concludes our work on learning perceptual reward functions.

6.1 Rationale and related work

We draw inspiration from how humans and animals acquire new skills to capture the many required details involved in capturing task progression. Primates and humans are known to possess a mirror neuron system that is at the basis of mirroring actions and behaviour of other individuals (Gallese, Keysers, and Rizzolatti 2004). This idea has been transferred to the field of robotics (Argall et al. 2009) in which a robot can acquire new skills by imitating the behaviour of the demonstrator. However, learning to solve a task from experts is suspect to copying the exact manipulations of the demonstrator. This is due to the learning agent not understanding the essence of the task. Moreover, no guidance is available when the agent arrives in unseen areas of the state space. A final problem preventing transferring expert demonstrations across actors is the correspondence problem (Nehaniv, Dautenhahn, et al. 2002): the embodiment of the demonstrator often differs from the learning actor. For example, the kinematic chain of a delta robot differs significantly from a human arm. Consequently, learning from demonstrations requires a mapping between different morphologies. Hence, a task progression metric needs to be invariant to the actor executing the task (i.e., the correspondence problem) and needs to be able to generalize to unseen situations.

In order to construct process monitoring metrics from demonstrations that capture the task intent, we need to (1) learn task-relevant representations, (2) solve the correspondence problem, and (3) translate the representation to a metric that indicates task progression and solution quality. This task progression metric can then be used for process monitoring. Additionally, this metric can be used as a reward function in a reinforcement learning setting such that the task can be learned from environment interaction.

6.1.1 Applications of process monitoring

The primary application we target is acquiring manipulation skills through RL. RL requires a reward function to signal the task performance to the agent. Contrary to popular approaches that use demonstrations in an imitation learning framework, we distil a reward function instead of a policy from the demonstrations.

The second application domain we target is process monitoring in manufacturing systems. An important challenge for smart manufacturing systems is finding relevant metrics that capture task quality and progression for process monitoring to ensure process reliability and safety. Data-driven process metrics construct features and labels from abundant raw process data, which incurs costs and inaccuracies due to the labelling process. This cost can be prevented by having process metrics that are learned without supervision.

We will use the terms “reward function” and “task progression metric” interchangeably to denote these two applications.

6.1.2 Data-driven process monitoring in smart manufacturing systems

Modern manufacturing systems are becoming increasingly complex due to high requirements on process quality and economical incentives (Yin et al. 2014). In order to ensure the reliability and quality of the outcome of industrial processes, process monitoring techniques are utilized (Ge, Song, and Gao 2013). Among process monitoring methods, data-driven process monitoring methods are a popular approach as they do not require modelling complex physical processes and can conveniently be collected with sensors and cameras. Data-driven process monitoring methods are of relevance for smart manufacturing systems that collect data at high volumes and frequencies (Wang 2012). The availability of this data enables data-driven methods to train models for process monitoring and fault detection (Yin et al. 2014). Machine learning methods, in particular, have been used to discover valuable patterns in manufacturing data by manually constructing features (Pham and Afify 2005). This way, virtual sensors can be trained to estimate product quality and process metrics based on historical measurements of easy-to-measure process variables. For example, in (Li and Zhu 2004) the quality metric of a paper pulping process is inferred from chemical process features constructed from surrounding sensors. To avoid the need of manually engineering features, deep learning methods are becoming increasingly popular for fault diagnosis (Zhao et al. 2019). In (Wen et al. 2018), they show that a deep neural network can outperform traditional process monitoring methods on three widely-used datasets. Other work looks at directly inputting process images to the neural network. For example, in (Lyu, Chen, and Song 2019) flame images of a furnace are used for monitoring the combustion process. In (Janssens et al. 2018) infrared thermal videos are used as training data for a deep neural network to estimate the health conditions of rotating machinery. In (Lei et al. 2016) raw manufacturing data is converted to latent features learned by an autoencoder neural network. In practice, many of these applications assume the availability of process experts in order to carefully label the data (Wuest et al. 2016). However, this is costly and ambiguous to do for some domains. To avoid labelling data, existing work (Kang, Kim, and Cho 2016) uses semi-supervised learning to exploit both labelled and unlabelled data for predicting wafer quality during semiconductor manufacturing. Another work (Malhotra et al. 2016) avoids the need of labelling the remaining useful lifetime of industrial machines by compressing the input sensor data to a latent space using a recurrent neural network autoencoder. By reconstructing the latent space to a machine health index, they can match the resulting time series and use the reconstruction error to compute the health index used for estimating the system remaining useful lifetime. However, their method still requires finding example health index curves. We utilize a similar idea to leverage data under nominal operating conditions while borrowing insights from the learning from demonstration research in order to learn a task progression metric.

6.1.3 Learning manipulation skills from demonstration

In Chapter 2, we already reviewed the background material on learning from demonstration. In this section, we reprise the main findings.

Learning from demonstrations Learning from demonstrations is a prevalent domain in the robotics learning community. In the learning from demonstration survey of (Argall et al. 2009), a distinction is made between giving demonstrations and imitation learning depending on whether an embodiment mapping exists. In case the teacher executions are demonstrated, an embodiment mapping is implicit. In contrast, imitation implies that the correspondence problem needs to be solved. These definitions place our work as imitation learning from external observations: sensors external to the executing entity are used to train a learning agent that can have a different morphology. One instance of learning from demonstration is behavioural cloning, in which supervised learning is used to predict the actions an expert would do in a given state (Ross, Gordon, and Bagnell 2011). However, behavioural cloning methods are known to copy end-effector trajectories instead of understanding how actions relate to task performance. Moreover, errors accumulate when an agent takes a wrong action, which pushes him into an unseen part of the state space. A more general way to force the agent to attend to which actions increase task performance, is to learn the reward from demonstrations instead of the policy.

Inverse Reinforcement Learning RL is a domain that shares similar semantics with process monitoring: both require metrics indicating task progression and quality. In RL, the task progression metric is known as the reward function. This signal guides the learning agent towards task solutions. A sub-domain known as inverse RL (Ng, Russell, et al. 2000) deals with learning reward functions from demonstration. In inverse RL, an outer loop learns the reward function while the inner loop executes a learning procedure for finding an optimal policy given the current reward function. Recent methods have looked at integrating deep neural networks as a representation layer in inverse RL (Finn, Levine, and Abbeel 2016; Ho and Ermon 2016; Fu, Luo, and Levine 2018). However, much computational power is required for training due to the two loops taking place. Speeding up the training process with kinesthetic teach-in and updating instead of optimizing the reward function is explored in (Finn, Levine, and Abbeel 2016). Unfortunately, manually moving the robot's end-effector proves to be unfeasible for tasks with difficult dynamics like knot tying or folding clothing. Other methods (Ho and Ermon 2016; Fu, Luo, and Levine 2018) leverage expert demonstrations based on adversarial training. In these setups, the goal is to learn the task directly and not infer a reward function. In contrast, we aim to learn a reward function completely decoupled from policy optimization. This way, it can be used for multiple purposes, such as learning and process monitoring.

Self-supervised learning An emerging method for sample-efficient learning of task-relevant features is self-supervised learning. Self-supervised learning exploits the structure present in a dataset to learn rich representations used for a downstream task such as image classification. Both natural language processing and computer vision has seen large leaps in self-supervised methods with, for example, BERT (Devlin et al. 2019). The general idea is to provide an artificial task to learn meaningful representations. Example tasks are learning to colourize images (Zhang, Isola, and Efros 2016), reconstructing the original input (Pathak et al. 2016) and predicting the relative position of two random patches (Doersch, Gupta, and Efros 2015). An instance of self-supervised learning is contrastive learning, in which representations are learned by providing contrasting examples. In the case of video demonstrations, time can be used as a supervisory signal to provide contrasting examples. The goal then becomes to recover the temporal coherence of a video. One of the firsts works leveraging time as contrastive signal (Misra, Zitnick, and Hebert 2016) inputs a sequence of frames and classifies whether the frames are in the correct order. Later work (H.-Y. Lee et al. 2017; Fernando et al. 2017) also frames self-supervised learning as a classification task in which the correct temporal order has to be determined. Several prior works construct reward functions, or equivalently process monitoring metrics, in latent spaces trained with time as a supervisory signal.

Using time as self-supervisory signal has been used in multiple works for learning from demonstrations (S. Nair et al. 2020; A. V. Nair et al. 2018; Sermanet et al. 2018; Dwibedi et al. 2018; Hartikainen et al. 2019). However, prior approaches assume the possibility of teleoperation, or express the reward as distance in embedding space which is not possible when a trajectory of steps has to be followed to arrive at the goal state. This is in fact the main reason for us to innovate beyond using Euclidean distance in TCNs embeddings: the distance between start and goal state in an embedding space that encodes the state of textile is not useful when a sequence of states need to be visited to arrive at the goal state. Consider the hypothetical one-dimensional embedding that encodes the state of cloth in Figure 6.1. In this example, the initial state is the unfolded shirt and the target state is a folded result. To get to the target state, the agent needs to pass through the intermediate states of folding both sleeves. However, the agent has no incentive to move from the initial state as the reward, i.e. distance between current and target state, will decrease when trying to go to the intermediate states.

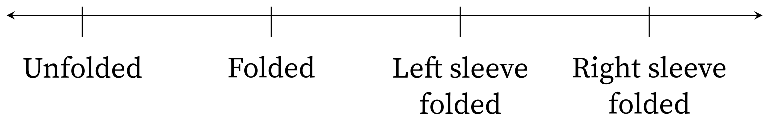


Figure 6.1 Hypothetical one-dimensional embedding that encodes the state of cloth.

Solving the problem of initial and target state being close in embedding space can be solved in multiple ways. For example, it is possible to tweak the loss function to force a certain structure in the embedding. An example of adding extra structure in the loss function is adding a cycle-consistency loss (Dwibedi et al. 2019) term. Cycle consistency means that a sample can cycle back to itself in embedding space when going to its nearest neighbour. For example, taking the nearest neighbor of sample $\mathbf{x}_i$ in embedding space ($f(\mathbf{x}_i)$) arrives at sample $\mathbf{x}_j$. Taking the nearest neighbour of sample $\mathbf{x}_j$ in embedding space should arrive back at sample $\mathbf{x}_i$. They demonstrate impressive results on aligning video pairs of an action recognition dataset. However, it is unclear how their method behaves on long video demonstrations containing multiple, possibly suboptimal and heavily out-of-phase solutions to achieve the same task. In our approach, we look more broadly at time series alignment methods.

Time series alignment In order to match the latent space trajectory of an expert demonstration to a learning agent, the latent space progressions must be compared. Given the presence of a time dimension, a time series alignment problem arises. Time series alignment is studied extensively in natural language processing (Myers, Rabiner, and Rosenberg 1980), bioinformatics (Seyler et al. 2015) and human activity recognition (Machado et al. 2015). Biological sequence alignment methods arrange the sequences of DNA to identify regions of similarity that may influence functional relationships. Path Similarity Analysis (Seyler et al. 2015) for example, quantifies the similarity and difference between protein transition paths.

Another broad class of algorithms for comparing a series of values with each other is Dynamic Time Warping (DTW). In DTW, the time series are assumed to be similar in amplitude but locally out of phase. DTW was introduced in (Bellman and Kalaba 1959) and had the goal to find an optimal alignment between sequences by warping the time axis iteratively. DTW has been used for a variety of domains such as speech recognition applications (Myers, Rabiner, and Rosenberg 1980), sign language recognition (Kuzmanic and Zanchi 2007) and time series clustering (Niennattrakul and Ratanamahatana 2007). Although many improvements on the original DTW algorithm exists (Folgado et al. 2018), we experimentally show that the canonical DTW with minor adjustments can be used to align the latent space progression between expert and learner.

6.2 Overview of the proposed framework to learn reward functions

We define the problem of using multiperspective images with task demonstrations to construct a metric indicating task progression and solution quality. We want the task progression metric to increase on important moments when progression is made towards solving the task. Central in our framework is (1) generating a meaningful semantic embedding that indicates task progression and solution quality and (2) mapping this embedding to a scalar value. A high-level overview is given in Figure 6.2. Our method consists of three main steps. First, we use multiperspective video frames to train an embedding using contrastive learning. This is done by using time as a self-supervisory signal. This method allows to learn useful invariances and forces the network to focus on task-relevant properties, as we will show in Section 6.4. We take a small sample of the demonstrations which we label *experts* as they will be used as a reference for indicating task progression. Second, we align the embeddings of the demonstrations to the task executions of experts using dynamic time warping. Third, we use this alignment to query the ensemble of experts for predicting task progress.

Our method assumes the availability of task demonstrations with corresponding process metrics. The demonstrations can range from teleoperated machinery to sensor recordings external to the executing body. We particularly focus on using recorded process metrics in the form of multiperspective camera streams. Our method is deployable for arbitrary processes and tasks for which example demonstrations are available. These demonstrations are allowed to contain sub-par solution strategies. The method requires selecting a small part of the data, in our experiments 5 %, as a reference for a good task solution. We focus on task

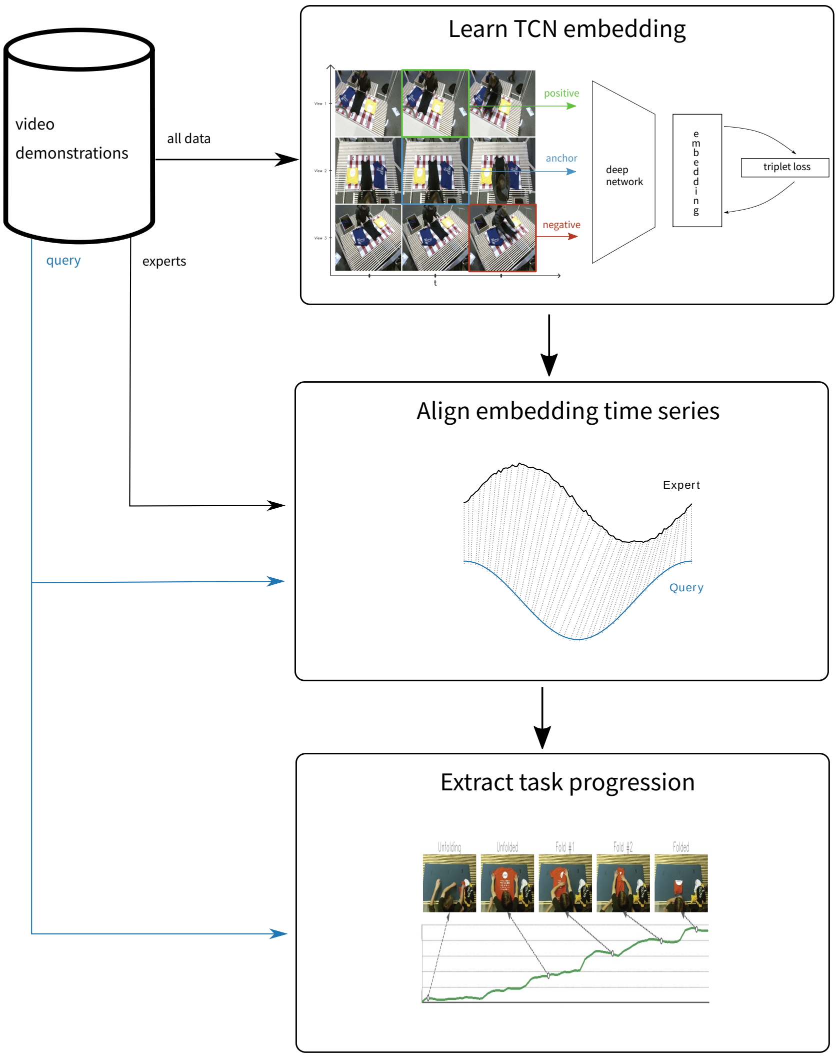


Figure 6.2 High-level overview of our methodology. We train an embedding by using time as contrastive learning signal. We then align all embeddings to a small set of high-quality demonstrations (labelled as experts). Finally, we extract a progression metric using an ensemble of the resulting alignment.

demonstrations given by humans, but any entity solving the task can be used as input. Our methodology is applicable in settings where process monitoring is essential for output quality. We exploit temporal coherence, which requires the process to contain measurable inputs along a temporal dimension. For example, multiple cameras filming how human workers are sewing the front and back of a shirt together. Another major application we target is learning robotic manipulation skills using RL. Our method allows learning a reward function from visual demonstrations, which can be used downstream for a learning agent requiring supervision in the form of a scalar value expressing task progression.

6.3 Methodology for unsupervised learning of reward functions

In the previous section, we gave a high-level overview of our framework to extract task progression metrics from crowd-sourced RGB images. Here, we discuss the framework in detail. We break down the three main steps into separate subsections.

6.3.1 Learning semantic meaningful embeddings using TCNs

Central in the proposed framework is learning task-relevant representations containing the notion of task progression and solution quality. We use Time-Contrastive Networks (Sermanet et al. 2018) in which time serves as a supervisory signal. TCNs are a self-supervised method for training abstract representations of the progression of a task. The core concept is to push video frames distant in time away and pull them together when they are near in time. Multiple cameras are used to capture several perspectives of the same demonstration synchronized. This principle is shown in Figure 6.3. Any pair of frames from different camera angles that co-occurred must be close together in embedding space. Frames from the same camera angle separated by time are forced to be distant in embedding space. This principle encourages the network to attend to high-level features relevant to the task. Attending to irrelevant background noise or low-level features would attract negative pairs from the same perspective and repulse positive pairs from different perspectives. This way, the correspondence problem (Brass and Heyes 2005) for imitation learning can be solved. In case the network tries to explain the visual difference between two temporal distant frames by looking at the demonstrator, it would pull the anchor and negative close together, leading to a higher loss. The only way to achieve a lower loss is by looking at task-relevant features: what is consistently changing in the scene that cannot be attributed to changes in viewpoint, lighting, occlusion, and background.

Formally, if the embedding of an input is given by $f(\mathbf{x}) \in \mathbb{R}^d$, the goal then becomes to put the anchor $\mathbf{x}_i^a$ and positive $\mathbf{x}_i^p$ closer in embedding space compared to the anchor $\mathbf{x}_i^a$ and negative frame $\mathbf{x}_i^n$ (Schroff, Kalenichenko, and Philbin 2015):

$$\left\| f\left(\mathbf{x}_i^a\right) - f\left(\mathbf{x}_i^p\right) \right\|_2^2 + \alpha < \left\| f\left(\mathbf{x}_i^a\right) - f\left(\mathbf{x}_i^n\right) \right\|_2^2,$$
$$\forall \left(f\left(\mathbf{x}_i^a\right), f\left(\mathbf{x}_i^p\right), f\left(\mathbf{x}_i^n\right) \right) \in \mathcal{T},$$

with α being the margin enforced between positive and negative pairs. $\mathcal{T}$ represents all possible triplets, i.e. all *anchor-positive-negative* combinations. The term $\left\| f\left(\mathbf{x}_i^a\right) - f\left(\mathbf{x}_i^p\right) \right\|_2^2$ is the squared L^2 norm of the distance between the anchor and positive in embedding space. The loss we are trying to minimize then becomes:

$$\sum_i^N \left[\left\| f\left(\mathbf{x}_i^a\right) - f\left(\mathbf{x}_i^p\right) \right\|_2^2 - \left\| f\left(\mathbf{x}_i^a\right) - f\left(\mathbf{x}_i^n\right) \right\|_2^2 + \alpha \right]_+ .$$

To gather the triplets $\mathcal{T}$, we use a semi-hard triplet mining strategy with an increasing difficulty level. The goal of this strategy is to guide the training process to focus on increasingly harder *anchor-positive-negative* triplets. We do this by first sampling random anchors and positive frames from all possible perspectives. Positive frames are temporal neighbours at a maximum ϵ frames sampled around the anchor. Frames further away are labelled as negatives. This principle is visible in Figure 6.3. For each anchor during training, we select the most difficult positive, i.e., where the distance between anchor and positive is the largest. For this *anchor-positive* pair, we calculate the distance between the semi-hard negatives and anchor. Semi-hard negatives are defined as contrastive samples to the anchor which are of moderate difficulty: the distance between *anchor-negative* pair is marginally larger than the distance between the *anchor-positive* pair. Intuitively, this corresponds with pushing the fail-cases out of the minimal distance range one by one, starting with the easiest. We define the cost function of the batch as the average loss scores overall anchor frames. We provide the pseudocode for our training procedure in Algorithm 2.

6.3.2 Aligning expert video embeddings with query videos

The TCN embedding trained in the previous section gives rise to a multivariate time series. Our goal is to compare the time series embedding of a demonstrator $X = (x_1, \dots, x_i, \dots, x_N)$ to that of a chosen expert $Y = (y_1, \dots, y_i \dots, y_M)$ in order to judge the quality of the folding demonstration. To calculate the alignment between these time series, we use Dynamic Time Warping (DTW) (Bellman and Kalaba 1959). DTW is an algorithm for measuring the similarity between time series under time distortions. It minimizes the effects of shifting in time by allowing elastic transformations of the time series, subject to time-normalization constraints. This allows accounting for nonlinear task execution rate differences between two demonstrations. In DTW, an optimal path P^* mapping time series X and Y are found:

$$P^* = \underset{\phi_x, \phi_y}{\operatorname{argmin}} \qquad \sum_{t=1}^{\mathrm{T}} d(x_{\phi_x(t)}, y_{\phi_y(t)}),$$

with $d()$ being a local distance function, for example, Euclidean distance. The alignment between the two time series is established through the mapping functions $(\phi_x(t), \phi_y(t))$. The warping functions are limited by certain constraints in order to be meaningful. Common warping constraints (Rabiner and Juang

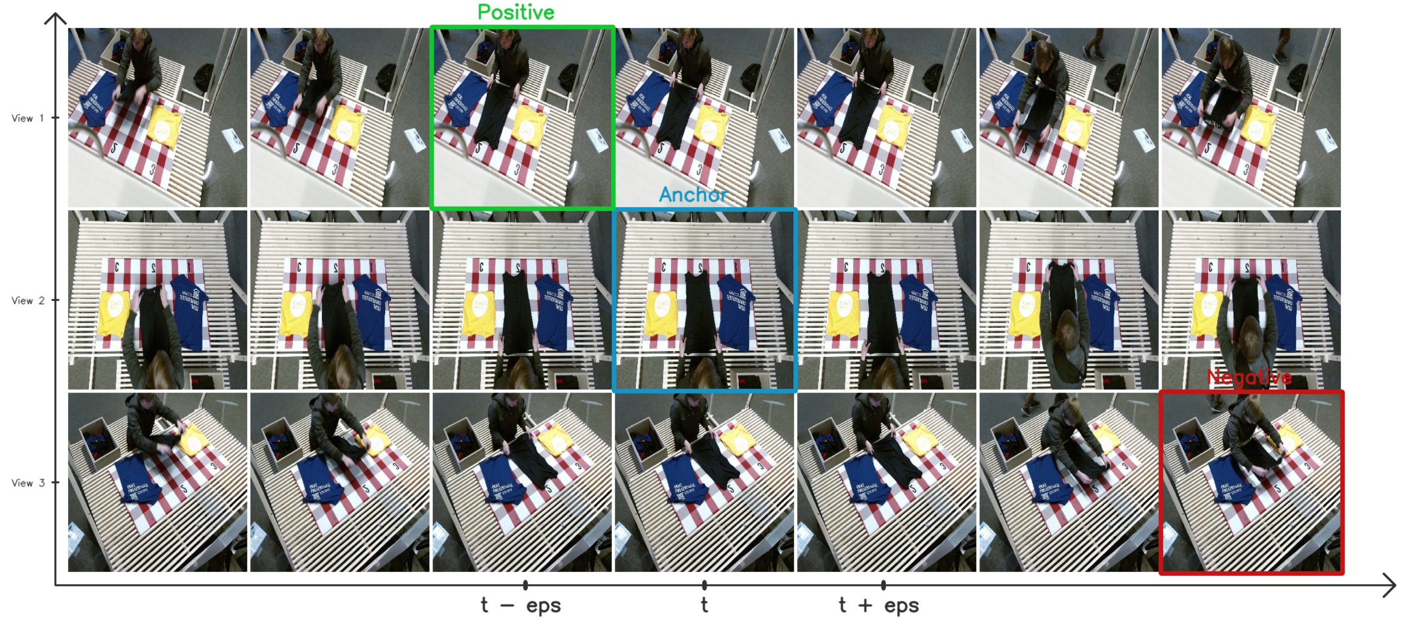


Figure 6.3 **Using time as a supervisory signal in TCNs.** A randomly selected anchor frame (in blue) and a nearby temporal neighbour from a different perspective (in green) are encouraged to be close in the embedding space compared to the anchor frame and a distant temporal neighbour (in red). This allows the network to learn to explain changes in the physical world.

Algorithm 2: Training loop of time-contrastive network

```
   Input: training set of videos $\mathscr{V}$,
          temporal distance $\epsilon$,
          neural network tcn parametrized by $\theta$,
          margin $\alpha$,
          mini-batch size $k$
1  foreach epoch do
2      loss = 0;
3      for 1 to $k$ do
4          select random video demonstration $v$ from $\mathscr{V}$ with
            frames $\mathscr{F}$;
5          select random anchor $a$ from $\mathscr{F}$;
6          generate positives
            $\mathscr{P} = \{p \in \mathscr{V} : \text{temporalDistance}(a, p) \leq \epsilon\}$;
7          find hardest positive
            $p^* = \text{argmax}_{p \in \mathscr{P}} \{\text{dist}(\text{tcn}(a), \text{tcn}(p))\}$;
8          generate negatives
            $\mathscr{N} = \{n \in \mathscr{V} : \text{temporalDistance}(a, n) > \epsilon\}$;
9          generate semi-hard negatives $\mathscr{N}_{sh}$ =
            $\{n \in \mathscr{N} : \text{dist}(\text{tcn}(a), \text{tcn}(p^*)) < \text{dist}(\text{tcn}(a), \text{tcn}(n))\} \cap$
            $\{n \in \mathscr{N} : \text{dist}(\text{tcn}(a), \text{tcn}(n)) < \text{dist}(\text{tcn}(a), \text{tcn}(p^*)) + \alpha\}$;

10         find easiest semi-hard negative $n^*$ with distance $\epsilon$
            from $a$ with $n^* = \min_{n \in \mathscr{N}_{sh}} \{\text{dist}(\text{tcn}(a), \text{tcn}(n))\}$;
11         loss +=
            $\text{dist}(\text{tcn}(a), \text{tcn}(p^*)) - \text{dist}(\text{tcn}(a), \text{tcn}(n^*)) + \alpha$;
12     end for
13     cost = loss/$k$;
14     Perform a gradient descent step on cost with respect to
        the network parameters $\theta$ of the neural network tcn;
15 end foreach
```

1993) are (1) start- and endpoint constraints as a clear start and end are manually specified by preprocessing the data, (2) monotonicity constraint which maintains temporal order during time normalization, and (3) local continuity constraints, also known as step patterns, to minimize loss of information. Compared to the canonical version of DTW, we relax the monotonicity constraint, which specifies that the alignment path does not go back in time. By allowing the time index of the expert time series Y (i.e., the reference signal) to go back in time, we can account for demonstrators restarting part of the task execution or even executing the task backward. This also allows for coping with a failed task execution. We also relax the formulation to open-end DTW (Tormene et al. 2009) by removing the endpoint constraint to incorporate the possibility that demonstrators are not finishing the task optimally or completely. A visual interpretation on how we align a demonstration to video frames of an expert is given in Figure 6.4. We show the single component of two fictitious one-dimensional embeddings. The demonstration of which we need to calculate the task progression is labelled as the query. This is aligned to the embedding time series of the expert. The warping path is drawn in the global cost matrix. The coloured sections on the time series and alignment path represent subtasks. The background colours represent similar task progression or the same subtasks in the videos of the expert and query sample.

6.3.3 Extracting task progression from embeddings

The last step in our methodology is to extract task progression indicators using the aligned embeddings. First, we search which frame of each expert aligns best with each frame of the demonstration. This is done by selecting the expert's frame with the lowest alignment cost. We then express the alignment cost as an alignment score by taking the reciprocal of the cost. Then, we select the temporal index of the best matching frame of every expert and express it as progress in percentage. Next, we average the experts' progress ratings by weighting with the normalized fit scores of the DTW phase of the pipeline. Finally, we remove any outliers by rejecting progress predictions of experts that deviate

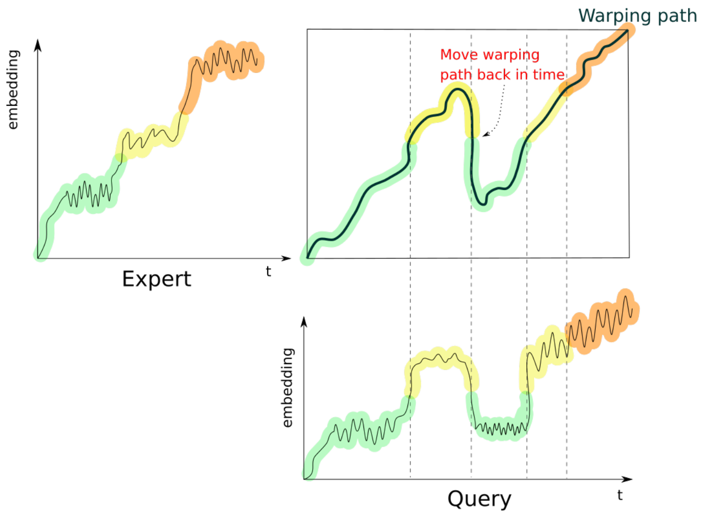


Figure 6.4 Dynamic Time Warping (DTW) without monotonicity constraint. We align a multivariate time series embedding of a demonstration (the query) to the video frames of an expert. The time index is allowed to go back in time in order to correctly align a demonstration in which executed actions undo the made progress.

too much from the median of the absolute deviations (Leys et al. 2013).

6.4 Results on folding clothing

We use the multi-perspective dataset of people folding clothing collected and processed in Chapter 5. We manually select 5 % of the data as experts, which will later represent the signal to align other video demonstrations using DTW. This selection was based on two simple criteria: (1) the resulting fold looks successful, and (2) the demonstrator executes substeps to solve the task in one go. The last criterion implies that to go from an initial state to a target state, the fold is executed in one flow and not paused in between or divided into multiple intermediate states. The data contains multiple sources of randomization as the recordings took place at two different locations, of which one is a public library. This makes it useful to learn multiple invariances.

We first discuss quantitative results on training the embedding and extracting the reward functions. Afterwards, we conduct a qualitative assessment by inferring what the embeddings are encoding and testing the reward functions for specific adversary cases.

6.4.1 Training results

The first step in our methodology is to train TCNs using multiperspective images of task demonstrations. The neural network architecture we used is given in Figure 6.5. We utilize the DenseNet (G. Huang et al. 2017) architecture pre-trained on ImageNet (Deng et al. 2009) as it already contains semantic relevant features for general-purpose vision-based tasks. Depending on the application, other neural network architectures can be used as a backbone. We append four trainable convolutional layers to the output of the DenseNet architecture. Afterwards, the output is passed to a spatial softmax layer (Levine et al. 2016). The spatial soft arg-max layer produces the expected image-space positions of the points of maximal activations of the features in the former convolutional layer. This allows decoding of the relative position of salient objects in the scene. The spatial softmax layer is succeeded by two fully-connected layers of 2048 and 32 neurons. The neurons in the last layer represent the compact low-dimensional representation of the task execution, which we will use downstream to generate a task progression metric. We train the network for 500 epochs using Adam optimization. Other fixed parameters during our experiments are given in Table 6.1.

To tune the hyperparameters of our method, we split the dataset into a training, validation and test set. We define the split at the level of the demonstration and not at the collective set frames. This means that a validation and test sample is a complete, novel example demonstration of folding a piece of cloth. We define a range for each identified hyperparameter in Table 6.2. Then, we evaluate all possible combinations of the hyperparameters using the semi-hard triplet loss, the number of semi-hard triplets and the fraction of successful hard negatives. We explain these quantitative metrics in the next paragraph. Finally, we select the set of hyperparameters that performs best on these training metrics on the validation set. The test set is used for the subsequent steps of translating the learned embedding to a scalar progression value. We provide the final hyperparameters used in our experiments in Table 6.2.

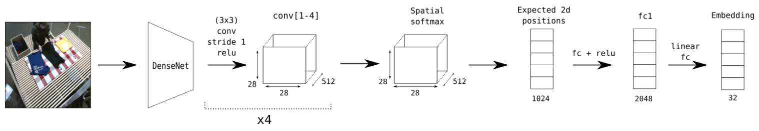


Figure 6.5 **Neural network architecture.** We use a pre-trained DenseNet backbone appended with four trainable convolutional layers. It is followed by a spatial softmax layer, which produces the expected 2D coordinates of the region of maximal activation in each channel of conv4. These coordinates are manipulated by a fully connected layer, which is finally passed through to the compressed embedding space.

Table 6.1 Fixed settings during training of the TCN embedding.

Hyperparameter	Value
Optimizer	Adam
Weight initialization	Glorot initialization
# Epochs	500
Image size	224×224 pixels
GPU	1 NVIDIA Quadro P6000

In Figure 6.6, we show quantitative results of training the TCN. The loss quickly and steadily improves, as visible in Figure 6.6a. This indicates that transfer learning from the DenseNet backbone advances smoothly. However, the semi-hard triplet mining strategy presents more difficult training triplets over time to the network. This might result in oscillating loss functions making it necessary to monitor additional metrics. In particular, we monitor the number of semi-hard triplets still available in the batches and the percentage of successful hard negatives. Figure 6.6b shows that pushing a new semi-hard negative out of the margin does not lead to an increased number of new semi-hard triplets re-entering the margin. Figure 6.6c examines which

Table 6.2 Hyperparameters used for learning an embedding to extract task progression metrics for folding clothing items.

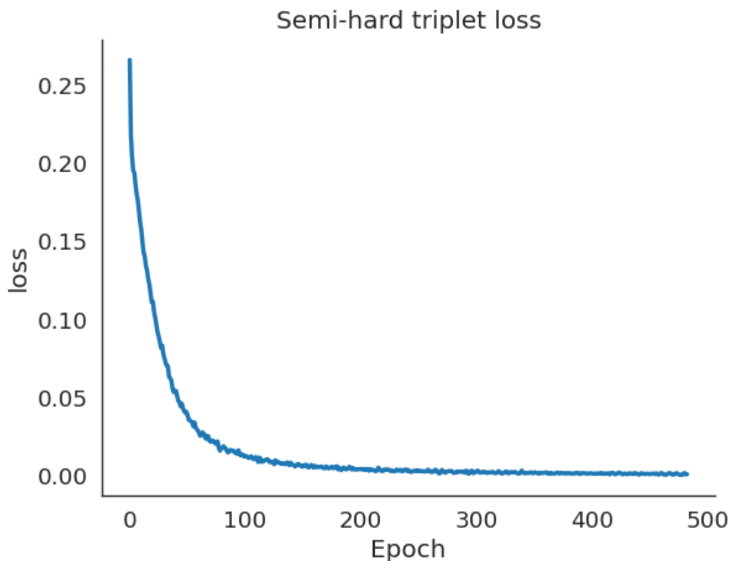
Hyperparameter	Best value	Value range
Learning rate	0.001	$[0.0001, 0.001, 0.01, 0.1]$
Batch size	32 samples	$[16, 32, 64]$
Embedding dimension	32 neurons	$[16, 32, 64]$
Max temporal distance between anchor and positive	3 s	0.5 s, 1 s, 3 s
Margin	0.2	$[0.1, 0.2, 0.4, 0.8]$
NN backbone	DenseNet	VGG, ResNet, DenseNet

fraction of the *anchor-hardest positive* pairs are closer in embedding space compared to the *anchor-hardest negative* pairs. This metric steadily increases, indicating that meaningful clusters are formed in embedding space to temporally separate images from different viewpoints.

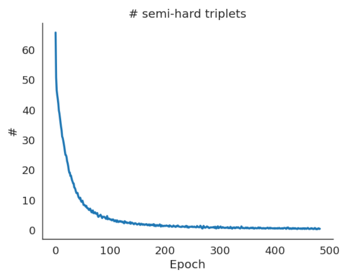
6.4.2 Reward function results

We align the resulting embeddings using DTW as described in Section 6.3.2. We use a symmetric stepping pattern (Rabiner and Juang 1993) with a maximum step size up to 10 frames, corresponding with approximately 1 second. Figure 6.7 shows the task progression, or reward function, of a sample in the dataset. We annotated points in the plot with the corresponding image in the video demonstration. This shows that the task progression metric increases at meaningful moments in the demonstration. For example, when the demonstrator grasps the shirt to unfold on the table, the progression score increases from 0 % to 40 %. Afterwards, the progression scalar value stagnates because the demonstrator slowly moves the right sleeve to the middle. Once the sleeve is folded, the progression indicator climbs. Next, the demonstrator grabs the left part of the shirt relatively quickly and moves it to the centre. This action is reflected in the progression value raising more quickly compared to the previous fold. Finally, the demonstrator receives maximum progression on completely folding the shirt. We find these results to be consistent among samples in the dataset. We provide videos of the task progression metric of other samples on `https://youtu.be/_HJhn8Hbv5s`. The whole pipeline requires 63 ms to process a single frame, with the slowest step being the alignment.

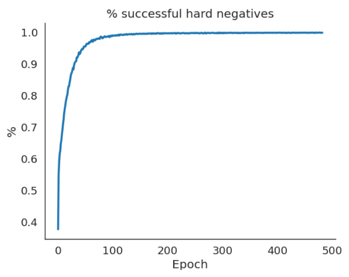
In conclusion, we find training of the embeddings in a self-supervised manner to be stable and efficient by using a semi-hard triplet function loss where we push out the easiest hard cases out of the margin first. Aligning the resulting embeddings with DTW on manually selected reference samples and using the alignment to express task progression increases progression scores on meaningful moments in the demonstration. This suggests that the embeddings encode task-relevant features.



(a) Semi-hard triplet loss



(b) Number of semi-hard triplets



(c) Percentage of hard negatives successfully separated

Figure 6.6 Loss and training metrics of the TCN training process with semi-hard triplet loss as optimization objective.

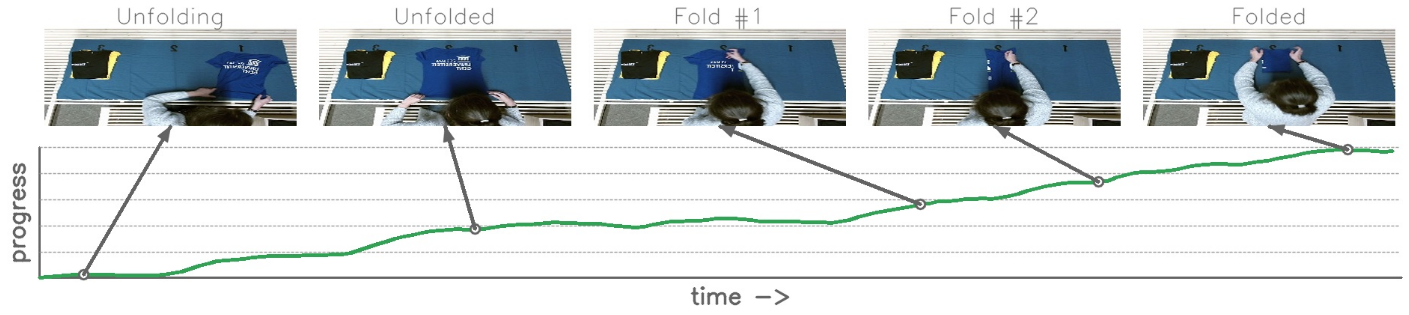


Figure 6.7 Task progression plot with corresponding video frames of a single demonstration. Each image is annotated with what is happening in the scene.

We analyze to which extent the embedding encodes important features for folding clothing in the next section.

6.5 Discussion

In the previous section, we have shown that TCN embeddings can be temporally aligned to extract useful task progression metrics. We now analyze the embeddings for post hoc interpretability. The goal is to discover which semantics are learned from the input images. However, neural networks lack decomposability into intuitive and understandable components. This is why we leverage the following two methods to understand what the network is encoding. First, we look at the semantic meaning of the embeddings at multiple temporal levels in Section 6.5.1. Second, we employ a case-based reasoning approach to interpret the learned representations and run robustness tests in Section 6.5.2.

6.5.1 Semantic Meaning of Learned TCN Embeddings

To discover the encoded semantics in the learned representations, we examine how the embedding progresses over time while linking it to what is happening in the scene. First, we examine how the embedding changes during the progress of a demonstration. To make the visualization interpretable, we project it to a lower dimension. Second, we examine whether we can extract meaningful information when searching for clusters in latent space.

To qualitatively analyze the results of training the embedding, we project the 32D embedding space to 2D using UMAP (McInnes et al. 2018). In Figure 6.8 we plotted the resulting projection mapped to the corresponding frame at different time shots. The time index is indicated with the colour of the scatter plot: from magenta to red, yellow, green, and blue.

A reoccurring observation is the projection of the embedding jumping at meaningful moments during task progression. In Figure 6.8, we notice that while grasping the shirt from the pile, the embedding stays in the first quadrant. Once the shirt is unfolded on the table, the embedding jumps to quadrant two. During the execution of the required folding steps, the embedding jumps to other locations. For example, the embedding gently transitions from quadrant two to three when folding the right and left sleeves. This suggests that the embedding can recognize different substeps in the folding task. When performing the final fold, the embedding is positioned between the first and fourth quadrant. We find this observation to be consistent among other samples. Consequently, the embedding at the start and the end of a trial is very similar. The explanation can be found in the observations starting and ending with a pile of unfolded shirts on the right side of the table and a pile of folded shirts on the left side. A similar start and end encoding in embedding space imply that the network potentially has problems distinguishing the start and end of the trial from a single image when there is no notion of memory.

Another observation is the embedding following a trajectory to go from grasping a piece of unfolded clothing to completely folding it on the table. Given that, in general, the third quadrant encodes folding the shirt's sleeves, it is possible that folding methods that do not fold the sleeves will not be assigned the correct progression score as there is no alignment available. In other words, for the DTW alignment to work, there needs to be a trajectory followed in embedding space in order to arrive at the solution. We explore this in Section 6.5.2.

To further analyze whether the embedding can distinguish meaningful moments during task execution, we generate clusters in the embedding hyperspace. We perform agglomerative clustering with ward linkage on each demonstration separately. Because our data is temporally sorted, we set the connectivity matrix as a square matrix with ones on the superdiagonal and subdiagonal, and zeros elsewhere. This way, we enforce the bottom-up cluster formation to consider temporal neighbours, reflecting the temporal ordering. We select five clusters to identify as they reflect the substeps in the task: grasping, flatten, fold one side, fold another side, fold in the middle. We visualize the results of the clustering in Figure 6.9. Here, we show the 2D

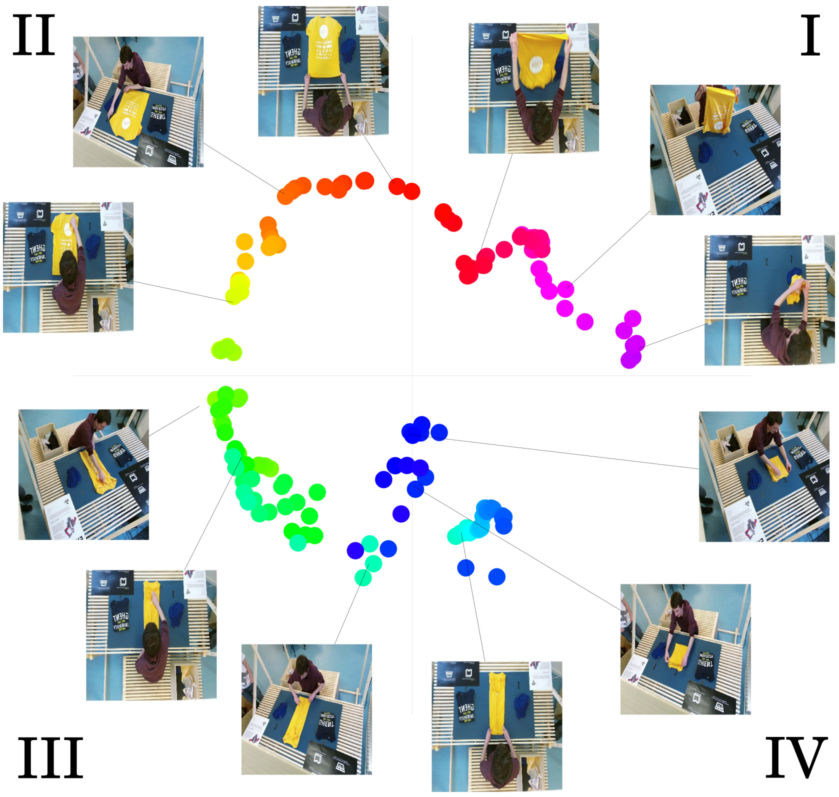


Figure 6.8 2D projection of the learned embedding. We annotate some points in the quadrants with the corresponding images to offer insight in to what is happening in the scene. Colors in the scatter plot indicate time progression from magenta to yellow, green and finally to blue.

projection of using UMAP on the embedding, with the colour representing the cluster membership instead of the temporal dimension. Qualitatively, we find the emergence of subtasks in the cluster membership when running agglomerative clustering in embedding space. The task starts in quadrant I in the projected embedding space. This is indicated with the red cluster membership. Once the demonstrator unfolds the shirt on the table, the cluster membership switches to green while the embedding transitions towards quadrant II. The same process repeats when transitioning between the other subtasks. This indicates that the embedding encodes relevant aspects of the task, which downstream algorithms can use. However, we have no guarantees that the network picks up other signals from the environment during training. For example, the network might be encoding the position of the hands, or the size of the shirt. We examine which features the network is attending to in the following subsection.

6.5.2 Case-based examples for post hoc interpretability

For the learned task progression metrics to be useful, they must be able to capture and generalize to specific situations that arise during the execution of the task, not seen in the training set. For example, a task executor can be performing random manipulations without actually performing meaningful contributions towards progressing the given task. This is also the case for learning purposes; many RL algorithms start with an exploration phase in which the robot acts randomly. To research the generalizability of our method, we employ a case-based approach in which we input certain scenarios and qualitatively analyze the result of our pipeline. We first describe the relevant scenario, followed by the resulting task progression scores, and offer insight into why a specific scenario succeeds or fails. Figure 6.10 and Figure 6.11 contains the visualizations of the cases we discuss below. We annotated the plotted task progression with specific frames from the demonstration to explain changes in the assigned progression. Videos of these hold-out samples are available at https://youtu.be/ZvK0pQWH8ec.

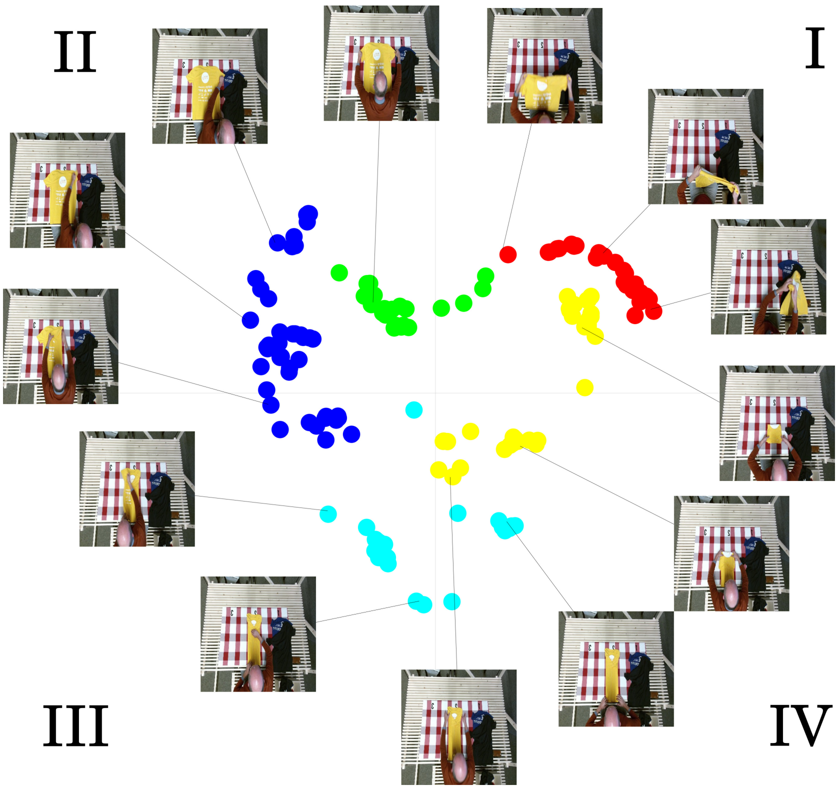


Figure 6.9 **Interpretation of embedding using agglomerative clustering.** Colors in the scatterplot indicate cluster membership. One point in the scatterplot represents an embedded video frame, projected onto a 2D plane using UMAP. The line links images to the corresponding embedding.

Change of environment with background distractions. Practical reasons and safety concerns make it possible that workers perform the designed process flow in another environment compared to the demonstrators. To test whether the learned progression metric can cope with this, we set up the folding table in a location not seen during training and fold clothing while putting random objects on the table. The images corresponding to the annotated parts of the plotted task progression in Figure 6.10a show that adding distractor objects such as a safety helmet, a flashlight, and a quadruped robot, do not prevent the learned task progression metric from assigning a correct progression score. There are two potential reasons for this. First, the data is recorded in a public environment where distractions are inherently present. Second, the TCNs are forced to attend high-level, task-relevant features. This filters out distractions in the image.

Meaningful manipulations compared to random behavior. Non-meaningful, random manipulations in the process execution occur in both process monitoring and scenarios where agents are learning to solve a task. Because fully random behaviour is not present in our crowdsourced dataset, it is uncertain how the learned progression metric associate this with process quality. We explore how the progression metric evolves by first doing a meaningful manipulation of the shirt, followed by randomly moving the hands above the shirt. This experiment is displayed in Figure 6.10b. We find that the progression metric correctly assigns an intermediate score on unfolding the shirt. For the subsequent random movement of the arms of the demonstrator, no additional progress is assigned. In similar experiments where demonstrators move their arms without any shirts on the table, we noticed that the network tries to distil meaningful manipulations. For example, a frequently reoccurring movement is performing the final fold where demonstrators grab the shirt at the bottom and fold it to the top. In some of those cases, we notice that the progression score increases due to the trajectory of the hands being recognized. However, the progression correctly drops when the network detects that there is no shirt being folded. We explore this further in the next paragraph.

Attention to the essence of the task. In the previous paragraph, we examined the effect of random behaviour on the learned task progression metric. Here we compare non-meaningful behaviour to meaningful manipulations to test to which extent the neural network is paying attention to the essence of the task. We do this by setting up a scenario to test to which extent the neural network looks at the hand trajectories while monitoring the state of the textile. Concretely, we try to fool the network: we first solve a subtask and then execute the required arm trajectories to solve the task but without touching the clothing. We hypothesize that in case the embedding is solely looking at the executed trajectories and not the clothing, the assigned task progression will increase. The result in Figure 6.10c. shows that the progression value increases when the shirt is unfolded. It is followed by stagnation of the progression score because the demonstrator is not manipulating the clothing. This demonstrates that our method looks at task-relevant features, like the state of the clothing, to indicate task progression. *This feature is what sets our method apart from behavioural cloning*; our method searches for the essence of the task instead of imitating end-effector trajectories.

Different task execution speed. The resulting progression metric should cope with different task execution speeds resulting from using different demonstrating entities. Given our crowd-sourced dataset, this issue is already present in the training data. It is solved by aligning the resulting embedding time series with DTW. We verify that this is working as intended by performing the folds of a shirt at different speeds. We provide the results of this experiment in Figure 6.10d. We label the start and end of the folds in order to indicate the different speeds at which the task is solved. For example, the first fold is executed rapidly, leading to a quick increase in task progression values. Contrary, the second fold is executed by moving the right sleeve very slowly to the centre of the shirt. Once the sleeve is folded, a sudden increase in task progression is given.

Different task executor morphology. Industrial processes can be performed by any human actor or machine. In order for the learned progression metric to be useful, it has to generalize across actors with different morphologies. We test this by folding the shirt with two people, such that four arms are manipulating the clothing. The results, visible in Figure 6.10e, show that the resulting progression metric behaves correctly. This demonstrates that our trained embedding is invariant to the actor executing the task.

Variations in task execution. Task progression metrics should cope with different variations in executing the task. We test this in multiple ways: we fold the task by lying out the shirt diagonally, undo, and repeat some of the substeps and doing excessive wrinkle flattening. We find that the given progression value is correct and consistent in all these scenarios. In particular, we examine the case in which a shirt is folded and then unfolded again. The unfolding is executed by the demonstrator and not by playing the video in reverse. The resulting task progression, visible in Figure 6.10f, shows that the maximum progression is reached when the shirt is folded. When the demonstrator starts undoing the fold step by step, the task progression correctly drops. This hold-out sample demonstrates that our alignment step does not contain an upward drift. This is important as the training data only exist of successful folds, leading to alignments that primarily run from start to end of the anchor and demonstrator. Hence the DTW step extracts useful information from the embedding to cope with failing task executions.

Generalization towards other folding methods. In extension to examining how well the learned task progression metric copes with variations on task execution, we look at how well they generalize when unseen folding methods are used. We test the case in which the demonstrator uses an alternative, unseen folding method where folds are executed on the table and in the air. In Figure 6.11a, we find that the assigned progression increases while folds are realized. We notice that the progression score suddenly drops on the step *unfolded square* (annotated in grey font above the corresponding image). By looking at the embeddings, as demonstrated in Section 6.5.1, we find that is due to the embedding state transitioning to the *unfolded* step. Afterwards, the progression correctly increases to the maximum level while performing the last folding steps. We also test the case when folding only two steps instead of four. The results of this experiment are visible in Figure 6.11b. In this case, the process monitoring metric only detects the folded shirt very late in the folding process. The explanation can be found in the training data: the experts have never seen folding solutions with four steps. Given that the alignment process expects a certain trajectory to be followed in embedding space, the alignment fails.

Generalization towards other instances of the target object. We introduce shirts with other colours, textures, and reflective material to investigate how well our method generalizes to other shirt instances. We provide an example demonstration of a shirt with reflective material in Figure 6.11c. Reflective objects are known to cause problems for object recognition neural networks (Sajjan et al. 2019). In this case, the learned process monitoring metric detects increasing progression on meaningful moments when solving the task. One instance in which the progression metric did not react appropriately is when folding tiny shirts, for example, in Figure 6.11d. When increasing the size of the shirt to normal proportions, the progression metric starts behaving appropriately. We hypothesize this is due to the embedding not reacting to very small hand movements and changes in the shirt, which lead to small pixel value changes in the image.

Task quality. We investigate to which degree the learned process monitoring metric incorporates the quality of the end result. We do this by examining how the task progression evolves when small and large disruptions are made in the resulting fold of the shirt. In Figure 6.11e, we disarrange the end fold of the shirt considerably. This leads to the demonstrator not achieving the maximum task progression at the end of the episode because the sleeves are partly hanging out of the shirt. However, in the experiment visualized in Figure 6.11f, we apply a small perturbation of the folded shirt, which is not picked up by the process monitoring metric. We find that a rectangular folded shape with many wrinkles receives the same progression score as a perfectly flattened one. This can potentially be solved by using cameras closer to the shirt, using depth information to incorporate wrinkles in the embedding, or manually constructing triplets of end results with good and bad flattening.

To summarize, we find that our method captures meaningful events in the task by looking at relevant features in the scene. By examining the assigned task progression values on the discussed hold-out samples, we show that the learned process monitoring metric is invariant to the demonstrator's morphology, background scene, execution speed and distractions. Our method is largely invariant to the shirt being manipulated, except that when the shirt gets too small, the resulting folds are not detected. Additionally, our method is not fooled by performing the expected arm trajectories without actually folding the shirt. We notice that the maximum progression value achieved during a successful folding demonstration consistently corresponds to the end-fold of the shirt. However, it is not possible to make meaningful comparisons in the end-quality of the fold of different demonstrations. We find meaningful reactions to highly visible disruptions in the shirt, but not to small wrinkles and small imperfections in the fold. In general, we find that the learned process monitoring metric effectively captures task progression and small degrees of output quality.

6.6 Conclusion

Learning the intention of a task from example demonstrations is an important step for process monitoring in manufacturing systems and reinforcement learning of robotic manipulation skills. In particular, evaluating the progress of folding clothing requires dealing with an infinite amount of states and occlusions caused

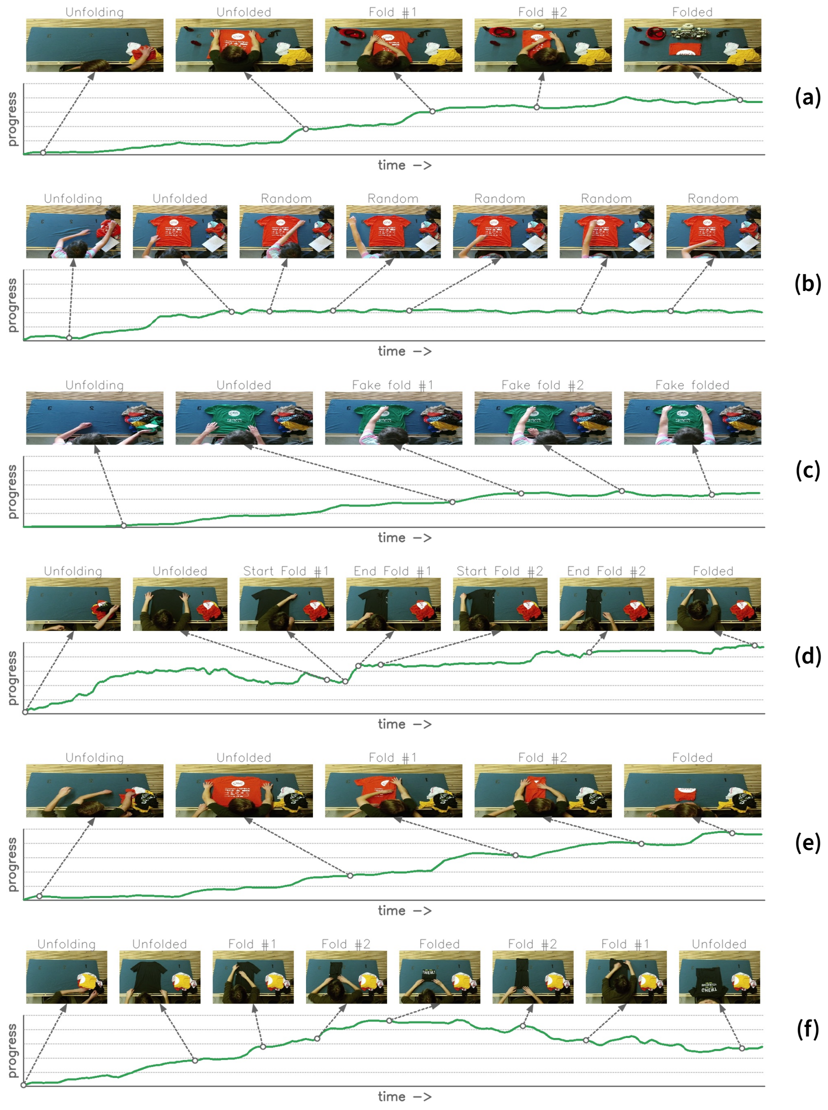


Figure 6.10 Task progression plots and corresponding images of out-of-sample cases to specifically test properties of the learned process monitoring metrics (part 1).

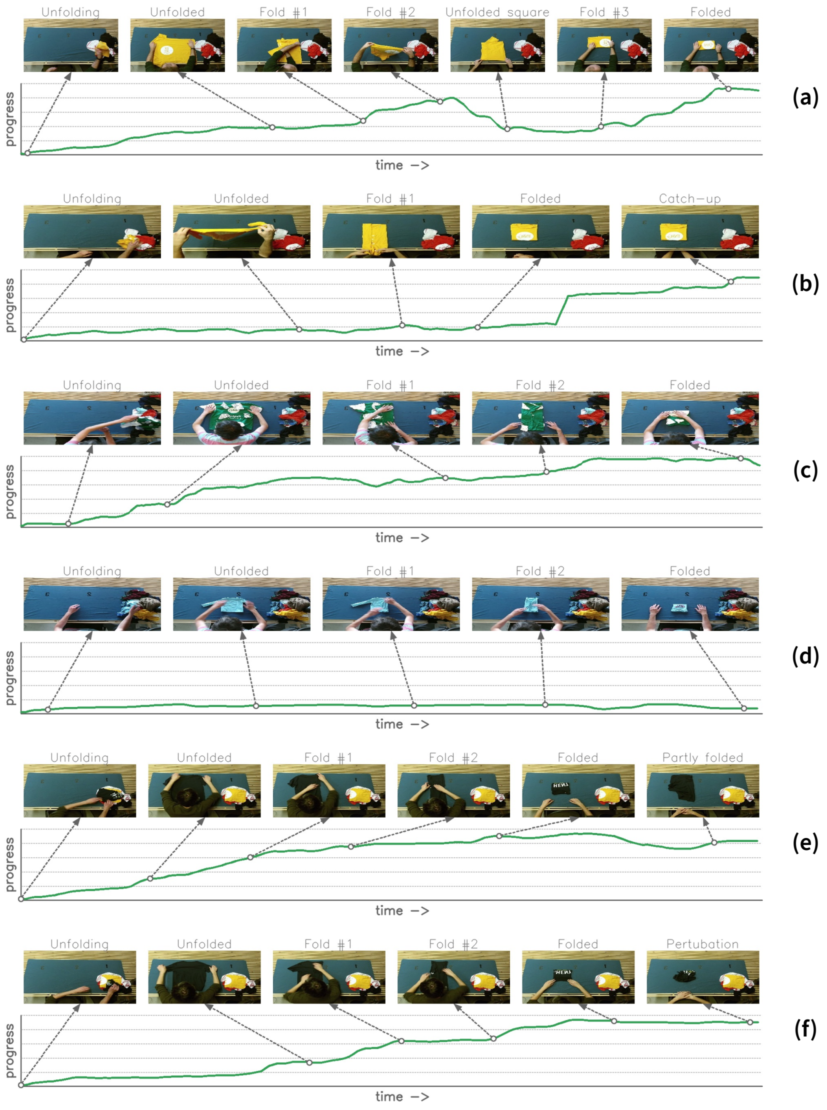


Figure 6.11 Task progression plots and corresponding images of out-of-sample cases to specifically test properties of the learned process monitoring metrics (part 2).

by deformations. While state estimation can be done with instrumentation of the cloth (Chapter 4), the requirement of translating the state to a scalar reward is still required.

In this chapter, we proposed a method to encode task intent by assigning a progression value during task execution. We do this by learning semantically relevant features by using time as a self-supervisory signal on videos with task demonstrations captured from multiple perspectives. We align the resulting embedding to express task progression and task quality. We used our method to demonstrate the first results on expressing task progression for the challenging case of folding clothing. We find that the process monitoring metric assigns correct progression values on meaningful moments during task execution. With case-based examples, we show that our method learns task progression metrics that are invariant to noise, actor morphology and execution speed. An important characteristic is that our approach does not require labelling task progression of existing demonstrations manually. Therefore, our methodology circumvents the need to engineer task progression metrics by learning the task intent from existing task demonstrations. Future work can explore the use of our learned reward function in a RL setting, incorporating multiple modalities, using probabilistic embeddings and exploring different loss functions. We discuss these ideas in detail in Section 8.2.3. In the following chapter, we also reflect on future work but from a high-level perspective, based on the experience of conducting the research presented in this book.

7

7 Towards learning robotic manipulation of clothing

Future general-purpose robots are multi-modal systems leveraging preprogrammed priors and data with embodied intelligence to understand and act on the world.

7 Towards learning robotic manipulation of clothing

In this dissertation, we considered the problem of learning robotic folding of clothing items by using simulation, smart clothing and learning task progression from human demonstrations. In this chapter, we zoom out to take a birds-eye perspective on the field of robotic cloth folding in order to highlight future areas of improvement. Our goal is to describe high-potential areas of research that, according to the research done in this dissertation, require further investigation for the field to advance towards learning robotic manipulation of clothing articles.

7.1 Improving sample efficiency

Data-driven approaches have triggered new developments for solving robotic manipulation problems. In the context of robotic cloth folding, we believe data-driven robotic perception and control will remain omnipresent to learn cloth dynamics, material properties and sensorimotor skills by interacting with the environment. In order to capitalize on learning-based methods, future research has to focus on improving the sample efficiency of these data-hungry methods. This need is imperative for the robotics domain, where experimentation with real robots is expensive.

7.1.1 Decouple end-to-end learning

Learn submodules instead of the complete system

End-to-end learning optimizes a task in unstructured environments with system imperfections by viewing the system holistically. However, it can lead to an immoderate amount of required training data. Hence, a first evident way to reduce data requirements is decoupling end-to-end learning into pipelines with dedicated modules. We have demonstrated deconstructing end-to-end learning in Chapter 4, where we used a smart cloth that autonomously communicates its state. This is in stark contrast to research that engineers vision pipelines (Wu et al. 2020) or incorporate learning the state implicitly in the model (Matas, James, and Davison 2018). The disadvantage of the latter approaches is potentially failing outside laboratory environments.

Another instance of decoupling the pipeline can be done by learning as much as possible offline. Batch reinforcement learning (Section 2.5) aims to maximally distil the knowledge in a static dataset. Theoretically, this is what Q-learning achieves: learn an optimal policy with suboptimal demonstrations (Sutton and Barto 2018). However, the deep variants have shown to be susceptible to performance collapse when only off-policy data is used (Hausknecht and Stone 2016). Current research in batch RL is ongoing and merits high potential for reducing the amount of data required for learning. Using batch RL is also a way to commit to the appeal to make learning look as much as possible as supervised learning. Indeed, the same way that supervised learning has turned data into strong pattern recognizers, batch RL can turn data into strong decision-making engines.

The last implementation of learning offline is shown in this research (Chapter 6), where we learned a semantically meaningful representation for the task using demonstrations. By generating pseudotasks, the agent can self-supervise its own learning process and acquire task-relevant representations without requiring real robot time. We elaborate on learning and using representations in Section 7.7.

Do not learn every submodule

Ultimately, *not every part of the pipeline has to be learned.* The data requirement of learned systems can be circumvented by incorporating available prior knowledge about the task. The body of knowledge described in Section 2.2 can be used as a prior for learning. For example, when learning from interaction to achieve wrinkle-free folding, the canonical way of defining the reward is a sparse signal indicating whether the cloth is folded or not. However, when end-to-end learning was not prevalent in the robotics folding community, researchers exploited gravity as a way to unfold clothing and remove wrinkles (Doumanoglou et al. 2016; Maitin-Shepard et al. 2010). This prior knowledge of exploiting gravity can be structured into the reward or task by driving the agent towards lifting the piece of cloth in the air.

Besides incorporating task knowledge, analytical models can improve the required learning time. For example, the work of F. Zhang et al. (2015) and our work in Chapter 3 define the action space in joint space, with each action being a movement of one joint with a fixed delta angle in each direction or remaining stationary. However, in both cases, the inverse kinematics of the robot is known, which implies that the action space can be defined as delta movements of the end-effector in Cartesian space. In the case of a 7 DOF arm, this example reduces the options in action space from 21 options (7 joints each having 3 options) to 6 options (forward, up, right, and the reverse options). Additionally, the robot does not require learning the inverse kinematics when an external module already solves it. However, this example would be difficult to realize if the kinematics are ill-defined, for example, when the robot arm contains a large degree of kinematic redundancy. For example, our pneumatic gripper implementation in Figure 4.13e is a heavily underactuated system.

7.1.2 Curriculum learning

A second approach for improving sample efficiency is adjusting the task instantiation to the skills already acquired by the agent. This principle of gradually ramping up the task difficulty is known as *curriculum learning* (Selfridge, Sutton, and Barto 1985). Different methods exist for introducing curricula. In robotic folding, it is possible to first learn how to fold stiff shirts by, for example, modelling planes joined with hinge joints. This is similar to the flip-fold device illustrated in Figure 7.1. Once the first task is learned, some deformability in the planes can be introduced and eventually scaffold the difficulty to a real shirt. Despite adding domain knowledge about the task difficulty can accelerate learning, finding which domain parameters are important to scaffold requires research and fine-tuning per domain. This ambiguity makes learning the curriculum a promising alternative. Hindsight experience replay (Andrychowicz et al. 2017), for example, maps states to goals and uses heuristics to carefully select which goal-trajectory sequences to replay. Other work proposes to learn the curricula via self-play to generate goal states (Sukhbaatar et al. 2017) or randomize the domain (Raparthy et al. 2020). Despite demonstrating successful results, the scope of tasks considered in the former work is limited to artificial environments containing a small state space. There is, to the best of our knowledge, no automatic curricula generating method that is able to overcome the instability associated with multiagent systems where both agents are learning in complex environments.

The two former approaches of using curriculum learning seemingly indicate an incongruity between manually-specified and fully-learned curricula. In reality, setting the curriculum is a spectrum in which elements of both manual and adaptive specification appear. Rudin et al. (2021) for example, learn quadruped locomotion by using a curriculum heuristic that adapts the slope and stairs in the environment based on how far the agent has learned to walk. We believe combining heuristic curricula with adaptive and manually-tuned environment elements avoids the instability and premature convergence associated with fully-learned curricula. In fact, this observation is the overarching theme; *we advocate there is a spectrum*

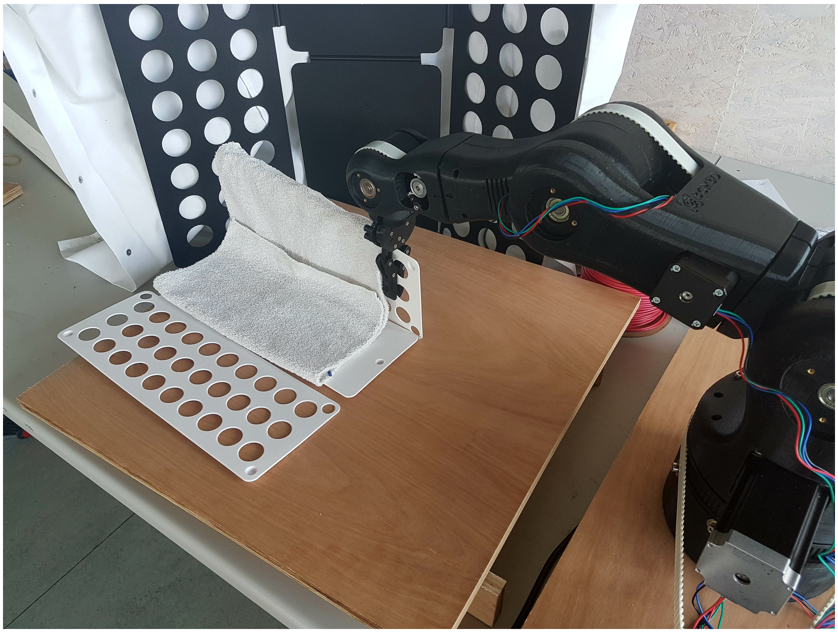


Figure 7.1 First learning to fold a flip-fold can be considered as an instance of curriculum learning towards learning to fold cloth.

to be leveraged between programming and data. Finding the balance between engineering and learning the curriculum is an important way forward to accelerate learning, enable Sim2Real transfer and uncover the aspects of the task that makes learning difficult.

7.1.3 Learning from demonstrations

Other potential gains for accelerating learning can be found by *leveraging demonstration data in RL*. Learning from demonstration has a long-standing history in robotics (Argall et al. 2009). Using demonstrations usually incurs one of the following problems. First, learning from demonstration is known to suffer from generalization and exploration issues in the case of behavioural cloning and dataset aggregation (Ibarz et al. 2021). Second, it is not always possible to collect a dataset of demonstrations. For example, recent work for combining RL and imitation learning insert kinaesthetically (Vecerik et al. 2018) or teleoperated demonstrations (Y. Zhu et al. 2018). However, teleoperating or moving the robot arm manually can be difficult for tasks where dynamics are important, like cloth folding or throwing darts. This is why we investigated the use of demonstrations external to the body of the robot in this research. However, using external observations as bootstrap for learning requires solving a correspondence problem between demonstrator and learner. For this reason, we believe that further gains can be found in the avenue of using external demonstrations as self-supervised representation learning method to be used for downstream tasks. We discuss possible technical future work of our method in Section 8.2.

An alternative form of providing demonstrations is in the form of supervision using a scripted policy. Residual RL, in particular, exploits the spectrum between programming and data by learning a policy that is additively combined with a scripted policy. The residual approach retains a scripted component which allows pushing the agent to the successful and important regions in the state space. In cloth folding for example, a hand-engineered controller can navigate the agent towards the cloth using simple segmentation methods that were discussed in Chapter 2 of this dissertation.

7.2 Datasets

Among computational resources and algorithm-centric advances, the availability of large amounts of data was important for deep learning to break through at the turn of the century. Datasets have played a similar role for the success of data-driven methods in robotic manipulation. In the case of learning to grasp objects, Dex-Net (Mahler et al. 2017a) puts forward a methodology to frame grasping as a supervised learning problem which leads to a qualitative learning signal. Dex-Net relies on generating a massive *dataset in simulation* containing 6.9 million images. This size is approximately seven times the size of the largest real-life data collection setup of a single institution containing robotic grasping attempts (Levine et al. 2016). In contrast to rigid object manipulation, there is a dataset scarcity in the domain of cloth folding. The publicly-shared cloth datasets by researchers are primarily vision-based. Augmenting such datasets with tactile sensing is of importance for dealing with occlusions in cloth manipulation. The field could advance forward with Dex-Net-like datasets containing a data generator with multiple objects having different physical properties. A recent initiative towards such datasets is executed in I. Huang et al. (2021) where grasp metrics for volumetric deformables are generated in simulation. Similar implementations for grasping and manipulating cloth could drive the field significantly forward for learning manipulation strategies and understanding cloth dynamics.

Although synthetic data is easily available using a simulator, it is plagued with transferability problems. These problems emphasize the importance of researching realistic simulations, robust algorithms and system identification. Using *real datasets* would alleviate these problems, but datasets in the field of cloth folding are rare. Datasets containing multiple modalities of different types of cloths and materials could help learning representations and understanding cloth interactions. Inspiration can be found in the retail fashion that contains high-quality datasets (Liu et al. 2016; Ge et al. 2019; X. Zou et al. 2019). Unfortunately, the modalities and item interaction are limited, making it non-trivial to find possible application avenues for robotic manipulation tasks. A future direction consists of applying the methodology FashionAI (X. Zou et al. 2019) uses to scale up the data collection and labelling. FashionAI relies on experts to disentangle attributes and label a small fraction of the data. Then, they train a network to label the remainder of the dataset. Training is halted when the network makes too many errors which could jeopardize the label quality. Such an approach can be used for labelling and understanding human interaction with cloth.

Real datasets containing example demonstrations of the task, can be used for bootstrapping learning. We filled in this gap with our research by providing a dataset of humans folding clothing ((Verleysen, Biondina, and wyffels 2020), Figure 7.2). A large-scale dataset with physical robot-cloth manipulations, similar to

(Levine et al. 2016) does not exist at the time of writing. Ideally, future efforts are directed towards gathering multimodal data of cloth interaction, possibly enriched with large-scale simulated data similar to Dex-Net. This dataset can then be used to understand the cloth's material properties, bootstrapping learning from the example demonstrations and learning semantically meaningful representations.

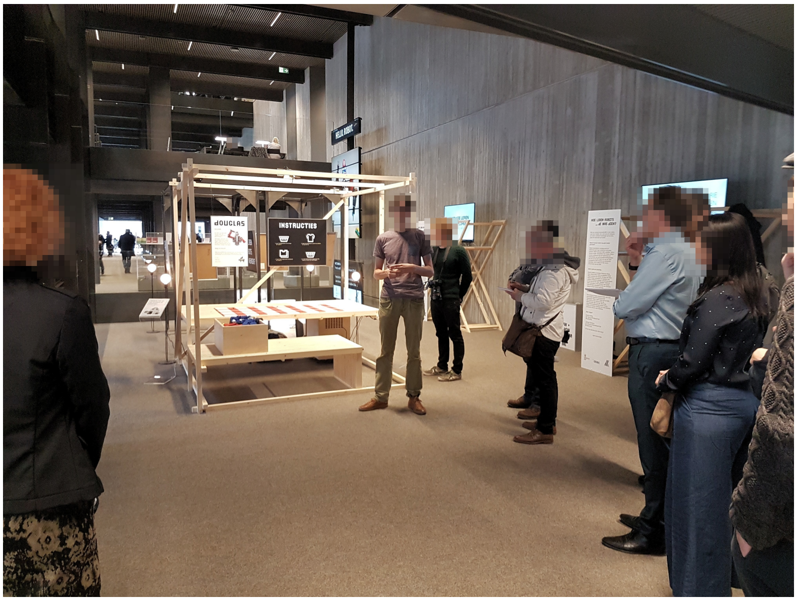


Figure 7.2 Towards letting robots learn from humans: our folding table setup to crowdsource data. A citizen science approach to crowdsourcing data can minimize researcher's bias in datasets. The folding table provides structure in the observations between demonstators.

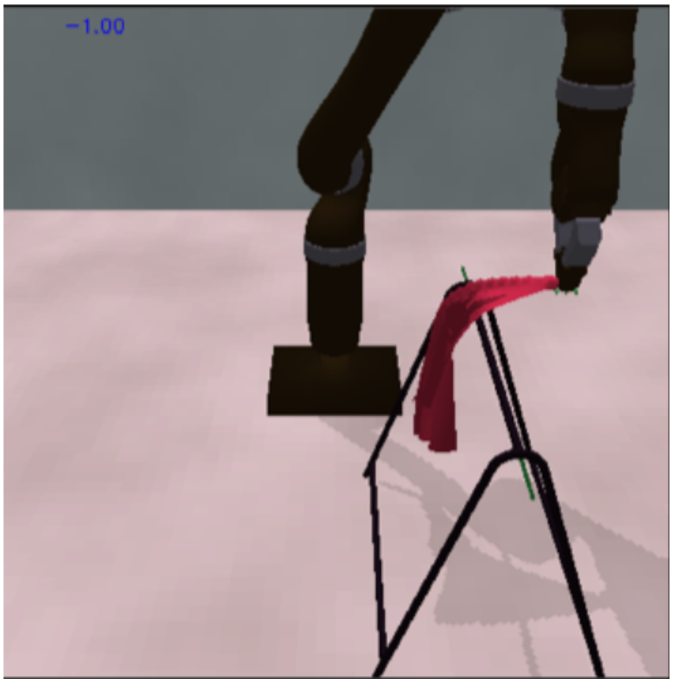
Finally, *benchmarking* robotic manipulation is an important step towards progressing the field of object manipulation. Benchmarking manipulation is challenging due to the diverse set of robots, object availability and costly robot interaction time. An additional difficulty is caused by the deformable nature of cloth: the cloth can be initialized in an infinite amount of starting configurations which makes it hard to start the task from a common initialization. A significant step towards filling this gap has recently been addressed in the work of (Garcia-Camacho et al. 2020) and Garcia-Camacho et al. (2021) where a standard set of objects, tasks and protocols are introduced for benchmarking common cloth manipulation tasks ranging from folding cloth to partial dressing. Future work can expand on this by including different clothing articles into the object set, considering multiple solution strategies for tasks like folding and extending the task set to cases where multiple objects are in the set. Concerning the evaluation of task performance, the field could use calibrated environments and software tools to measure task performance in a standardized way. Alternatively, our work in Chapter 6 potentially integrates with benchmarks as we provide a methodology to extract a task progression metric. The work of Garcia-Camacho et al. (2020) additionally proposes evaluating the performance of the entire system. However, given that the state perception of cloth is of fundamental importance, we find an ImageNet-like dataset for cloth currently missing in the community. This perception cloth dataset should contain multiple types of clothing articles, diverse configurations, interactions and heterogeneous input modalities. Ideally, benchmarks and datasets containing real objects are enriched with a simulated counterpart, calibrated on the real data. Given the success of simulation in robotics, researchers could run quick experiments in simulation and transfer results to the real world. These benchmarks fundamentally help the field forward as it brings best practices, standardization and comparison of experiments into the complex field of robotic cloth manipulation.

7.3 Simulation

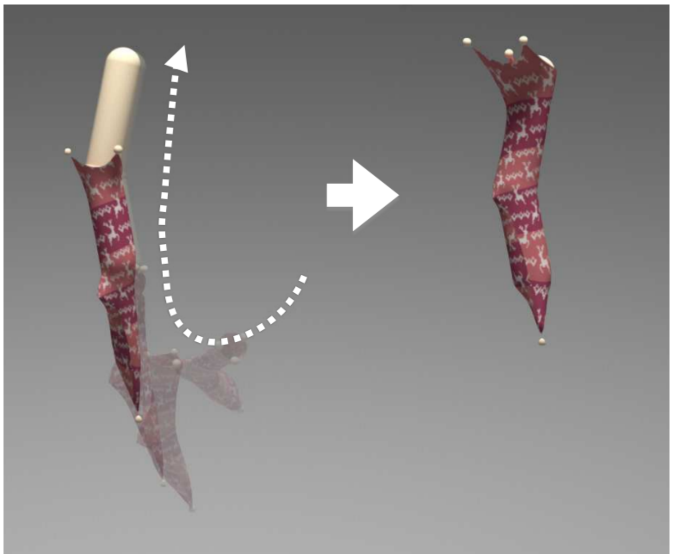
Data-driven methods from the machine learning domain have proven extremely powerful for adaptive and robust control. However, they require many examples which are expensive to generate on real, physical systems. The outlook remains that highly parameterized functions, such as neural networks, will keep on requiring and improving on large amounts of data (C. Sun et al. 2017). Hence, simulation remains an important tool for robotic learning to generate large datasets. As distilled in Chapter 2, there are roughly two roads for exploiting simulations: learning from interaction or constructing large datasets with realistic grasping examples, similar to Dex-Net (Mahler et al. 2017a). As of today, it appears the second road, i.e. generating datasets, has proven to be commercially exploitable in comparison to learning from interaction. Dex-Net, for example, has been commercialized two years after its advent[1]. Pursuing a similar path for robotic cloth folding contains merit but first requires further research and development in soft body simulators. Specifically, realistic simulation of friction, material deformation and other physical properties will have to be further developed before the deformable object manipulation domain can follow along the same path as its rigid counterpart. The outlook is promising as existing and new simulations started supporting soft body simulation, for example MuJoCo 2.0, Isaac[2], and SoftGym. In addition, state-of-the-art research in cloth simulation recently addresses real-time simulation of realistic cloth and solves inverse control problems like human-assisted dressing and material properties estimation (Liang, M. Lin, and Koltun 2019; Yifei Li et al. 2021). However, the support for robotics integration is limited. It appears that current robotics researchers have to trade-off simulation fidelity and cloth features versus integration possibilities with robotics and control. Hence, future directions should consider a tighter integration between high-fidelity cloth simulation and practicable robot control. For example, the cloth simulator in Figure 7.3a models no explicit contact dynamics but has integration with a robotics simulator. The more realistic cloth simulator in Figure 7.3b models contact dynamics but has no robotics simulator integration.

An important characteristic of the cloth simulation is the degree of parallelization. Running many learning environments in parallel has shown to allow learning quadruped locomotion on a single machine within a single day (Rudin et al. 2021). We believe this is a strong example that advocates for future cloth simulations to run on GPU in order to run learning environments massively in parallel. Our research addresses this idea with our cloth simulation on GPU developed in Chapter 3.

1. https://www.ambirobotics.com/
2. https://developer.nvidia.com/isaac-gym



(a) Source: (Matas, James, and Davison 2018)



(b) Source: (Yifei Li et al. 2021)

Figure 7.3 Towards more realistic cloth simulation and contact dynamics. From grasp implementations using constraints (Figure 7.3a) to realistic modelling of contact dynamics (Figure 7.3b)

Deformable bodies require reasoning about the shape, dynamics and material properties of the object. This makes neural networks a viable replacement for analytical models in simulations. Neural networks can learn the forward dynamics of cloth from sensory input. Moreover, forward passes through the network can be faster than a forward pass through all simulation substeps. However, dataset bias is an issue when training machine learning models to predict physics dynamics: world models are trained on a dataset, making it is unclear how well they generalize. Complex, learned dynamics models might show visually unrealistic deformations, lose volume over time and cannot deal with occlusion, which is bound to happen when manipulating with an end-effector (Mrowca et al. 2018; Yunzhu Li et al. 2018). Therefore, future research efforts should deal with occlusion by incrementally learning the object properties. Once again, insights from analytical models of cloth can be exploited in parametrizing the learning model. Graph neural networks, for example, reflect the interconnected particle nature of mass-spring simulation approach to cloth and show promising results for realistic and efficient cloth simulation (Pfaff et al. 2021). Alternatively, residual physics uses learned physics on top of an analytical model to predict the error in the forward dynamics (Golemo et al. 2018) or learn fine-grained, unmodelled effects in the simulator from data (Heiden et al. 2021).

Finally, differentiable simulations allow backpropagating gradients through the physical consequences of actions. Differentiable cloth simulation has been shown to be an effective approach for control problems and estimating material properties (Liang, M. Lin, and Koltun 2019; Yifei Li et al. 2021). A future avenue worth exploring is the co-optimization of body and brain: how would the robot morphology evolve if the parametrization of the end-effector is taken into the optimization loop. Outsourcing intelligence from the brain to the body is a principle known as morphological computation (Pfeifer 2006) and is highly visible in nature. For example, legged animals have their muscles arranged in a way that enables them to use simple neural signals for control (MacKay-Lyons 2002). This principle of outsourcing computation to morphology has been shown to be transferrable to quadruped locomotion by using compliance (Urbain et al., 2021). The same idea can be used to study how gripper embodiment and control can be co-learned to handle a variety of objects. This research would answer questions about which end-effector designs are optimal for manipulation of cloth and how different material properties lead to different gripper designs. For the application of cloth manipulation, it has already been shown that the choice of gripper design influences the complexity of the task dynamics (Borràs, Alenyà, and Torras 2020a). In order to optimize for such a co-design objective, a differentiable simulation can be used. By making the simulation and physics differentiable, gradient-based optimization can be used as an alternative to evolutionary strategies or RL. Gradients provide a direction in which to change the control parameters locally, leading to much quicker convergence (Degrave et al. 2019; Yifei Li et al. 2021). We prospect that by simultaneous evolving body and brain, new and innovative gripper concepts with competitive grasping performance will arise.

To summarize, we strongly advocate for further developing the realism and parallelism of cloth simulations that allow for semi-realistic robot-cloth interactions. Ideally, the simulator exposes numerical or analytical gradients which allow gradient-based optimization of inverse problems and robot morphology. Nonetheless, the question remains on how far we can drive the realism of the simulation. A high-quality cloth simulation of Blender, for example, is shown to be not transferrable to the real world in the work of (Tanaka, Arnold, and Yamazaki 2018). Hence, Sim2Real transfer remains an important research direction which we discuss next.

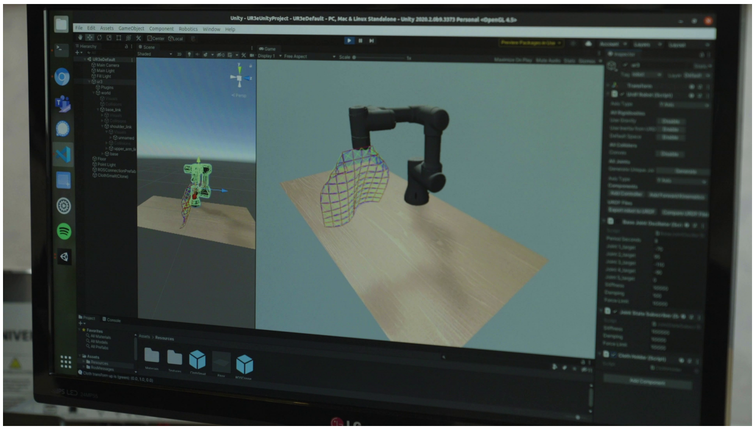
7.4 Sim2Real

Transferring skills learned in simulation to the real world will remain relevant because further increasing the accuracy of simulators like Bullet and Mujoco(cfr. Chapter 2), and making more robust controllers alone will not bridge the Sim2Real gap (Figure 7.4). In literature, there are a lot of success stories of domain randomization as a method for training robust controller in simulation. However, domain randomization, or more broadly simulation randomization, samples environment parameters uniformly which leads to demanding computational resources. Further research should pinpoint important environment parameters to vary and impose a curriculum for efficient learning. Another useful approach is system identification in which the physical parameters of the simulation are tuned. The discussed differentiable programming approaches for simulation in Section 7.3 merit potential in this regard.

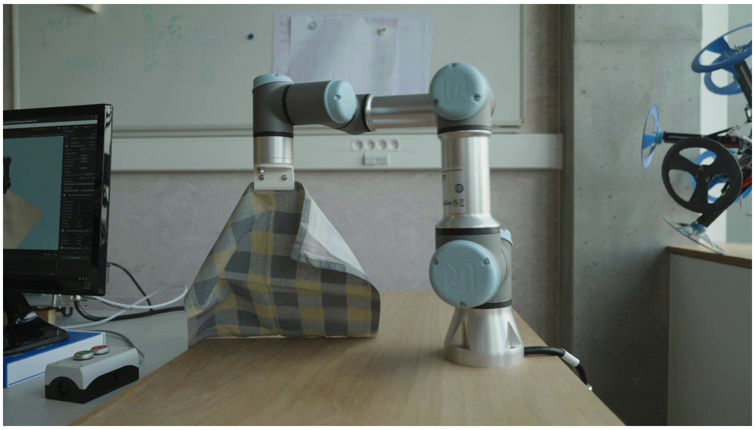
As a final note, cognitive sciences deem it plausible that humans operate under an intuitive understanding of physics rather than constructing exact physical models of reality (Baillargeon et al. 2010). This raises the question of how much importance should be dedicated to Sim2Real solutions that focus on tuning the simulation towards the real world as compared to developing models and representations that interpret physics and act on the understanding of these dynamics. We discuss sensing the environment and building representations in the following sections.

7.5 Grippers for robotic folding

Billard and Kragic (2019) bring to the attention that most robotic applications utilize parallel jaw grippers that are unable to solve tasks requiring dexterous manipulation. Similarly, Siciliano, Khatib, and Kröger (2008) note that dexterous, multi-fingered hands have not really been applied to any major application due to reliability, complexity and cost. Recently, Borràs, Alenyà, and Torras (2020a) brought a similar observation to the domain of cloth manipulation that most prior work uses a pinch grasp with both general-purpose and cloth-specific grippers. Indeed, the robot-cloth manipulation community is uncertain about what types of end-effectors are needed for optimally grasping and constraining cloth-like objects. For this purpose, Borràs, Alenyà, and Torras propose a framework to characterize grasps and manipulation primitives which has given rise to



(a) UR3 robot arm and cloth in simulation.



(b) UR3 robot arm and cloth in the real world.

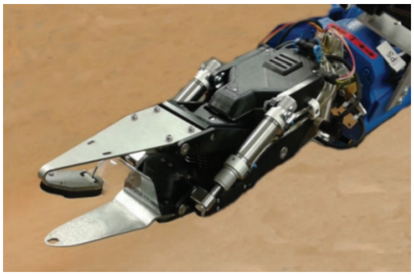
Figure 7.4 Sim2Real illustration with a UR3 robot and cloth. There is an accurate mapping between robot state by using proprioceptive sensors. Mapping the state of the cloth in simulation and the real world is an active research topic (cfr. Section 7.6)

novel gripper designs in (Donaire et al. 2020) for performing cloth-related tasks. The next step towards designing beyond anthropomorphic hands requires replacing rigid components with soft elements. Compliant elements lead to easier control and more safety. Soft-bubble (Kuppuswamy, Alspach, et al. 2020) for example, uses air-filled finger membranes for simultaneous compliant control and tactile sensing. These developments align with the philosophy that intelligence can be outsourced to the embodiment. As discussed in Section 7.3, we can extend these insights and methodologies by co-optimizing embodiment and control. We have argued that simultaneous optimization of body and brain is a research direction containing a lot of potential for optimizing the gripper shape for the robotic cloth manipulation task at hand (Figure 7.5). Differentiable programming allows computationally efficient co-design of embodiment and control, which will be the ultimate step towards truly effective multifunctional grippers.

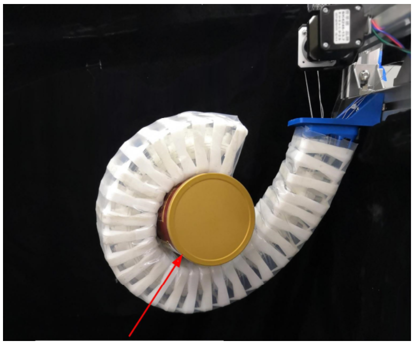
7.6 Sensing

Deep learning has proven to be a strong representation learning method that builds complex concepts out of primitive representations. The bulk of the robotics manipulation research that uses deep learning, leverages vision as input modality for training a deep neural network. While camera images are useful for extracting a global perspective on the shape of the object, capturing small-scale deformations in noisy environments with occlusions and changing lighting conditions is difficult. Moreover, sensing and reasoning about contact forces with the environment remain a problem for data-driven controllers. This is why future research should capitalize on acquiring and fusing multiple modalities. The work of (M. A. Lee et al. 2020) introduces a solution by learning a multi-modal latent space for robotic control tasks. In the deformables domain, tactile information is being used for estimating cloth properties (Yuan et al. 2018). However, a large barrier to using tactile sensing is the sensor hardware accessibility. Siciliano, Khatib, and Kröger note that “tactile sensing always seems years away from widespread utility compared to vision”

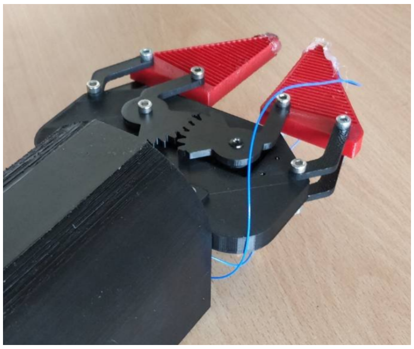
(a) Source: (Twardon and Ritter 2015)



(b) Source: (Doumanoglou et al. 2016)



(c) Source: (L. Li et al. 2019)



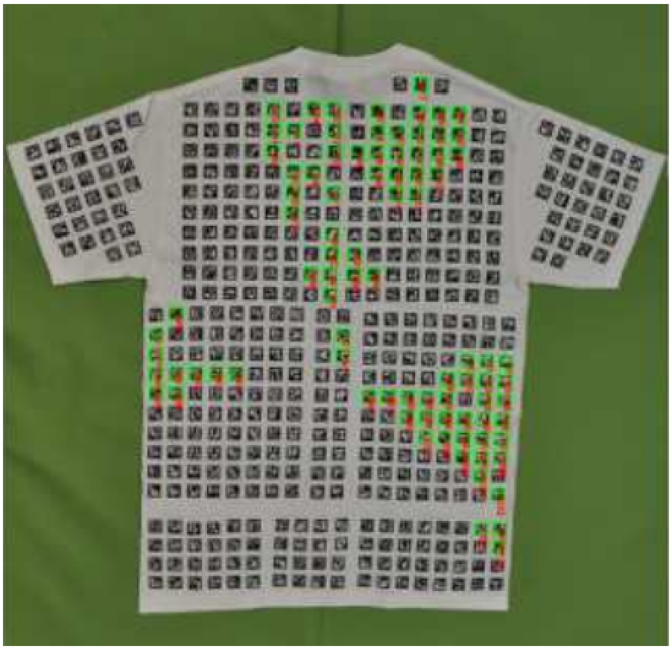
(d)

Figure 7.5 New designs for general-purpose (Figure 7.5a) and task-specific grippers (Figure 7.5b) for folding can be found by co-optimizing the morphology with control (for example, Figure 7.5c shows a pneumatic gripper based on the morphology of an seahorse and our gripper in Figure 7.5d is based on the fins of fish).

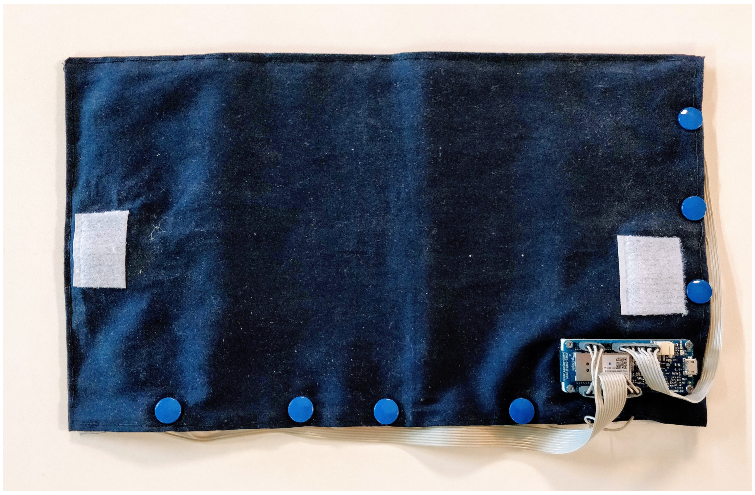
(Siciliano, Khatib, and Kröger 2008). While many types of tactile sensors exist, for example GelSight (Donlon et al. 2018) and FingerVision (Yamaguchi and Atkeson 2017), it is difficult to create a sensor that is inexpensive, has a fine-grained resolution with high sensitivity and is easy to integrate and use. Nonetheless, research recognizes the need for tactile sensing with cost-effective off-the-shelf sensors like Digit (Lambeta et al. 2020). In addition, sensors like Soft-bubble (Alspach et al. 2019), where the sensing is intrinsically part of the gripper's morphology, allow both high contact area grasping and high-resolution sensing. Future robots are preferably equipped with similar soft fingers containing tactile sensing by default.

Another aspect of tactile sensing is the integration of sensors into the object, a process we labelled *instrumentation* in this dissertation. We explored this in Chapter 4 by creating a smart textile embedded with tactile sensing that can communicate the state of the cloth (Figure 7.6). Future work should focus on increasing the type of forces being measured by embedding pressure, shear, strain and IMU sensors among others. The sensing hardware preferably resembles a sensor *skin* where instrumentation takes place on the surface of the object while minimizing the impact on material properties. This implies that the electronics inside the soft skin should be able to deform to irregular shapes. Inspiration for soft sensor skins can be found in the field of humanoid robots where for example iCub has embedded tactile sensing (Tomo et al. 2018). Our outlook here is that research in stretchable circuit technology allows further miniaturization for soft skin instrumentation of objects. The soft sensor skin in turn allows using compressed representations for learning downstream tasks on the real platform.

A missing piece of the instrumentation puzzle is *generalization from instrumented objects to non-instrumented objects*. Similar to the human ability to estimate the outline and texture of an object using touch only, cross-modal prediction models can learn to see by feeling and vice versa. Generally, cross-modal prediction models aim to predict data in one domain from the other. This is an active research domain in the vision and NLP domain in which they seek to generate captions for images, for example (Donahue et



(a) Source: (Bersch, Pitzer, and Kammel 2011)



(b) Source: (Verleysen et al. 2020)

Figure 7.6 Instrumentation facilitates state estimation and can reduce the required training time to acquire manipulation skills. A multidisciplinary approach goes beyond the vast focus on vision pipelines (instrumenting the shirt with visual markers Figure 7.6a) by integrating, for example, tactile sensing in cloth (piezoresistive sensing in cloth Figure 7.6b)

al. 2015). In robotics, cross-modal generative models have been used to generate an image based on tactile data and vice versa (Yunzhu Li et al. 2019). We postulate that similar methods can be used to learn representations that contain invariance to input modality. We elaborate on these ideas, based on our research, in the concluding Chapter 8.

7.7 Representations

A general question is how to generate appropriate representations from a rich, high-quality and multimodal dataset. Our work has demonstrated that useful representations can be learned by using contrasting examples that are mined in a self-supervised manner. Therefore, we support the idea that applying contrastive representation learning methods on cloth datasets lead to rich representations that are useful for downstream tasks. A remaining question is where to insert the representation in the cloth manipulation pipeline. Our work has used a smart cloth (Chapter 4) and cloth folding representation (Chapter 6) in the reward function. An open question remains how to avoid a learning agent exploiting learned reward functions. For this, we draw inspiration from the residual RL and residual physics domain. We hypothesize that *residual reward functions*, an additive reward function of an engineered and data-driven component, could prevent the agent from exploiting the reward function. Alternatively, the learned representation can be used as a state variable. This promising direction is, for example, explored in CURL (Laskin, Srinivas, and Abbeel 2020) where contrastive representations are optimized jointly with the RL objective.

A final outlook concerns the fundamental question on the appropriateness of deep neural networks in robotics and, more specifically, in RL that is characterized by incremental learning. Neural networks tend to forget what has been learned when new patterns are acquired. Although replay buffers are used in order to mitigate catastrophic forgetting, approaching neural networks differently on a conceptual level are more promising in our opinion. For example, memory-augmented neural networks represent memory explicitly and train a controller module that learns to store and manipulate memory (Graves, Wayne, and Danihelka 2014). An architecture with explicit memory can, for example, learn and retrieve different folding strategies depending on the situation. Furthermore, research should explore beyond connectionist methods. An example is given in the work of Jia et al. (2019) who demonstrate that tree-based methods can still be competitive in an era dominated by deep learning. Their work considers cloth manipulation tasks by exploiting the non-parametric nature of random forests to dynamically change the number of leaf nodes based on the given imitation data and new cloth configurations.

7.8 Future outlook on the field of robotic cloth manipulation

The field of robotic manipulation of clothing items has evolved from highly-engineered pipelines to data-driven methods over the past decade. The impressive results of RL prompted the community to shift from one end of the spectrum, i.e. fully-decomposed sub-systems, to the other end, i.e. end-to-end pipelines. Recent trends show that researchers discovered the limitations of end-to-end RL and are now finding the centrum of this spectrum. This centrum involves renewing the attention on hardware. *Democratizing hardware* developments allow fast and broad adoption by the community and smaller research labs that do not contain budgets to, for example, buy a $400.000 PR2 robot and equip it with $165.000 Shadow Hands to contribute towards learning bimanual dexterous manipulation skills. This is why future hardware development efforts should prioritize accessibility in terms of reproducibility, price and open-sourcing the implementation, similar to our instrumentation work of Chapter 4. For robotic cloth manipulation, the *importance of hardware* has recently been revived with a framework for understanding the relation between grasp types and gripper for cloth. A major insight that helps in this regard is outsourcing intelligence to the hardware embodiment for robust control. By *co-optimizing the body and brain*, the chain is closed and task-effective gripper designs with integrated control arise. Using *soft materials and integrated sensing* will play a vital role in the fabrication of grippers as clothing items have notoriously difficult state estimation. To facilitate state estimation, we foresee an important role for *instrumentation* towards smart cloth using robot sensing skin from stretchable circuit technology. We hypothesize that the resulting gripper and smart cloth implementations should be used for the *generation of a multimodal dataset* and benchmarks containing cloth-robot interactions. Ideally, such datasets contain a virtual counterpart calibrated on the real data. However, we believe that simulations should not pursue 100 % realism. Rather, the community should focus on developing models and representations that help to understand and act on physics in order to reduce the Sim2Real gap. The multi-modalities of the simulated and real data can be used to *learn meaningful representations* of deformations for downstream tasks. These tasks can be factorized at a level that interactions among sub-problems are small and most of the complexity is handled by the sub-systems. Finding solutions for the tasks contained in the sub-systems requires striking a balance between programming and leveraging data. A central philosophy we deem important is the presence of a *residual* component that merges prior knowledge of analytical models with the data-driven components of morphology, simulation, imposed curriculum, reward signals and finally, control policies.

Ultimately, the progress in deformable object manipulation will be determined by the available hardware options and control algorithms in order to acquire a large repertoire of manipulation skills. A holistic view where hardware and morphology close the loop with control will allow evolving manipulators tailored to the tasks and use them to build a representation of how the world works. Robots can use that feedback to understand how actions influence the environment and learn to solve tasks by using human examples, instrumented objects and their own experiences. Such progress would lead to a comprehensive set of deformable object manipulation skills and, ultimately, general-purpose robots.

8

8 Conclusion

We summarize our main findings from using simulation for robotic learning, developing smart textile, collecting a dataset and learning perceptual progression metrics.

8 Conclusion

Deformable objects change shape on interaction. These changes lead to an infinite amount of configurations that need to be considered when manipulating deformable objects like clothing items. This is problematic as rigid body manipulation methods are not easily transferrable to objects that deform. A solution for dealing with real-world variability is using learning-based methods by interacting with the environment. However, deep learning, the dominant learning paradigm, requires tremendous amounts of data to work. Several issues need to be addressed to effectively employ machine learning methods. These issues range from dedicated hardware for handling cloth to accelerating learning with priors to avoid generating months of training data on real robots.

8.1 Research conclusions

In this research, we filled in gaps towards learning robotic folding of clothing items. We started our dissertation by providing a self-contained chapter introducing the fundamentals of robotic manipulation of folding cloth (Chapter 2). We have discussed methodologies, tools and approaches to building a cloth folding pipeline while identifying strengths, weaknesses and gaps. This exposition gave rise to the motivations behind this research: **solving the lack of data and quick learning methods for learning to fold cloth.**

As a first solution to accelerating learning, we developed a cloth simulation environment (Chapter 3). A salient feature of our cloth simulator is that it leverages GPU acceleration, similar to how deep neural networks profit from concurrent calculations on GPU. **Our robotic-cloth simulator successfully uses deep RL to learn to fold a cloth two times, sequentially**. However, transfer to the real world is non-trivial as the simulation uses the cloth's full state, which is inaccessible in the real world.

To make state estimation of cloth possible in the real world, we developed a smart cloth (Chapter 4). Our smart cloth employs off-the-shelve components and can be fabricated DIY and low-cost. It sandwiches a piezoresistive rubber between two layers of conductive threads, forming a grid of tactile cells. By recording a dataset of labelled cloth states, we demonstrate that standard machine learning methods can classify the cloth state. The main feature of our smart cloth is that it can serve as a state observation or reward function in an RL framework. In our experiments, we employ the smart cloth as a sparse reward function indicating whether the cloth is folded or not. Our results demonstrate that **our low-cost smart textile can learn to fold a cloth on a low-cost dual robotic arm platform in-vivo**.

Data is a critical element for building intelligent systems. Yet, the field of folding clothing is less endowed with datasets compared to its rigid body counterpart. For this reason, **we constructed a data collection setup to crowdsource a dataset of people folding clothing articles** (Chapter 5). The benefit of taking a citizen science approach to collecting our data is the diversity it generates in example demonstrations. Our dataset consists of roughly 300.000 multi-perspective RGBD images and is available online for researchers and practitioners to use. We labelled the frames to include pose information and annotated folding quality and substeps. With this dataset, we aim to fill in a gap in learning deformable objects manipulation, bootstrapped by human examples.

Learning from interaction with the environment, formalized by RL, is a potential way to acquire robotic manipulation skills as it shares principles with how humans learn skills. A major element is the reward function that signals how well an agent is solving the task. In the case of folding cloth, constructing a reward signal is non-trivial: it is required to measure the state of the cloth, including folds and wrinkles, and translate it to a scalar value. Although this engineering is possible with instrumentation (Chapter 4), it is difficult to express how well a cloth is being folded. The field of RL solves this problem by considering the inverse problem: instead of finding the optimal policy using a reward function, what is the reward function given some optimal policy. However, inverse RL follows a computationally expensive approach by optimizing policy and reward simultaneously. For this reason, we decouple policy and reward learning. We have proposed **a method for learning reward functions in the form of progression metrics without labelling any data** (Chapter 6). Our labelling-free approach is achieved by utilizing self-supervised learning on time as contrastive signal. We validated our method on the dataset collected in Chapter 5. Using case-based experiments, we have found that our method learns task-relevant features and useful invariances, making it robust to noise, distractors and variations in the task and shirts. The experimental results have shown that the proposed method can monitor processes in domains where state representation is inherently challenging, such as folding clothing items.

Finally, we have compiled our vision for future directions of the field of robotic folding (Chapter 7). We have stressed the importance of democratized hardware, sensing, data and the integration between morphology and control.

8.2 Future research directions

While Chapter 7 provided our high-level perspectives on the field of robotic folding, this section zooms in on specific improvements and future directions for the methods and contributions of our research. We categorize future work in the following categories: (a) extending our cloth folding dataset; (b) improving the scope of our instrumentation process; and (c) learning reward functions from demonstrations.

8.2.1 Extending cloth folding dataset

Chapter 5 introduced our crowdsourced dataset of humans folding clothing items. As future work, the dataset can become a general-purpose cloth manipulation dataset. This is possible by adding novel clothing types to the set, such as trousers and socks. We noticed in our reward function experiments (Section 6.5) that the embedding is unable to distinguish small perturbations in the folds. Hence, adding a detailed view on the cloth being manipulated is useful in better identifying wrinkles and small perturbations on the cloth. This can be done by placing cameras with larger focal length focussed on the cloth. Alternatively, instrumented cloth (Chapter 4) can be added to the mix to provide an extra source of modality.

Our dataset can also evolve to a benchmark for cloth state classification and action recognition. A classification task can be added as a category to our folding dataset. A classification task of state configuration or tactile sensing reconstruction can be defined by recording the same clothing articles in different configurations, optionally with instrumentation. The clothing articles can optionally be linked to a virtual twin, using the cloth simulation we developed (Chapter 3). Eventually, these inclusions can lead to the dataset becoming a general benchmark instead of the role it now plays, i.e. providing example demonstrations to bootstrap learning.

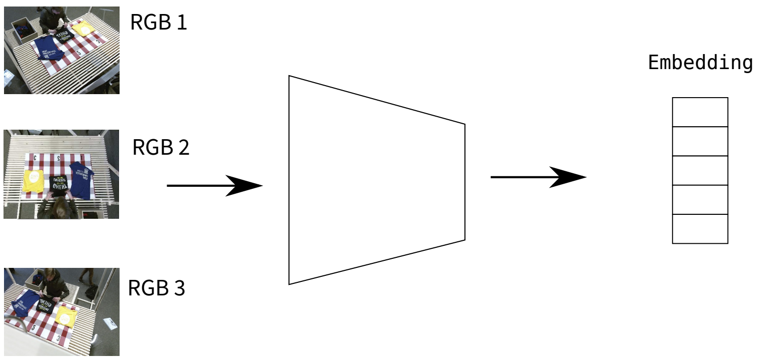
8.2.2 Instrumentation

The smart textile introduced in Chapter 4 was improved by Proesmans et al. (2022) to a modular, wireless sensing technology able to classify the state of larger cloths. With the availability of linking multiple sensing patches, future work can consider classifying more complex states, similar to the substeps defined in our dataset (Chapter 5), for example, left sleeve folded. An interesting follow-up is fusing the tactile information of the smart cloth with camera images in order to generate a multimodal database of self-labelled images of arbitrary cloth configurations. This data is produced by manipulating the cloth while recording its state with cameras. The multimodal dataset can then be used to learn a multimodal latent space as in (M. A. Lee et al. 2019). We elaborate on this idea in the following section.

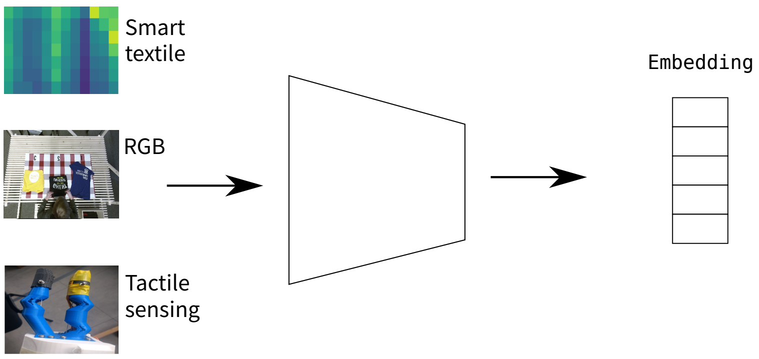
8.2.3 Learning reward functions from demonstrations

A key component in the reward learning framework proposed in Chapter 6 is the manner in which the embedding is trained. Training is done in a self-supervised way by building a dataset of anchor-positive-negative triplets from different perspectives using time as a contrastive signal. The central philosophy behind this training design is to learn useful invariances. By forcing two frames from a different viewpoint to be closer in embedding space than two temporal distant frames from the same viewpoint, we achieve viewpoint invariance. Similarly, we hypothesize that some form of input modality invariance can be learned by introducing different modalities. In this scenario, a camera image and tactile sensing of a cloth from the same timestamp should be closer in embedding space compared to the same camera image and the tactile input of the robot's finger from different timestamps. We visualize this idea in Figure 8.1. Another approach to including multiple modalities is similar to the approach in (M. A. Lee et al. 2019): different backbone architectures per modality fused into an embedding trained on pseudotasks like predicting optical flow and contact.

Generating meaningful representations when modalities can be dropped is an open research question. This question is of relevance for the smart textile we developed in Chapter 4: it is impossible to instrument all clothing articles. We believe our multimodal contrastive learning approach proposed above holds merit in dealing with missing modalities on certain clothing items. Our idea is to train a multimodal embedding with instrumented clothing and tactile finger sensing in the way described above. At test time, non-instrumented cloth sensing can be swapped by images as we hypothesize that the embedding is trained to become invariant to the input modality to some extent. Although this does not eliminate the difficulty estimating the state caused by occlusions of the cloth self-collisions, we believe that this training



(a) Three camera images from different perspectives as input.



(b) Multiple modalities as input.

Figure 8.1 Contrastive embedding trained with RGB images from different perspectives vs. trained with multiple modalities.

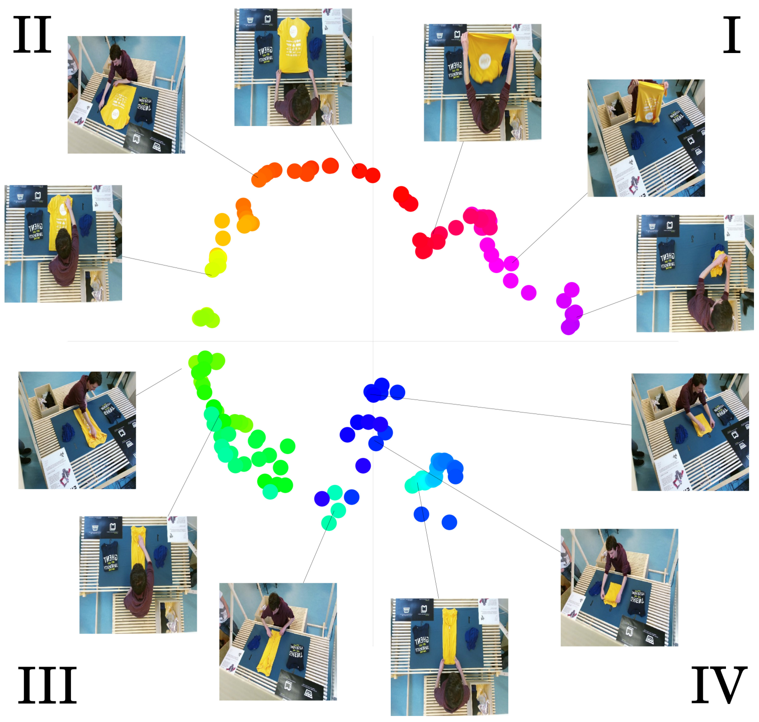
procedure can enable the network to estimate to which sensor reading it should belong in embedding space.

We demonstrated our methodology in the context of task progression metrics for folding clothing. A next application is to use our work on learning progression metrics as reward functions for RL tasks. However, neural networks are easily fooled with adversarial examples (Nguyen, Yosinski, and Clune 2015) and RL agents are known to exploit simulator and reward function dynamics. More generally, the problem of ensuring that a reinforcement learning agent's goal is aligned with our own goals is unsolved (Sutton and Barto 2018). However, we can aim to make our task progression metrics as robust as possible. One way to achieve more robustness is by adding a state machine to our framework, for example, by using HMMs. An alternative can be found in the work of Borràs, Alenyà, and Torras (2020b) where a representation of the environment is embedded in a graph of manipulation primitives.

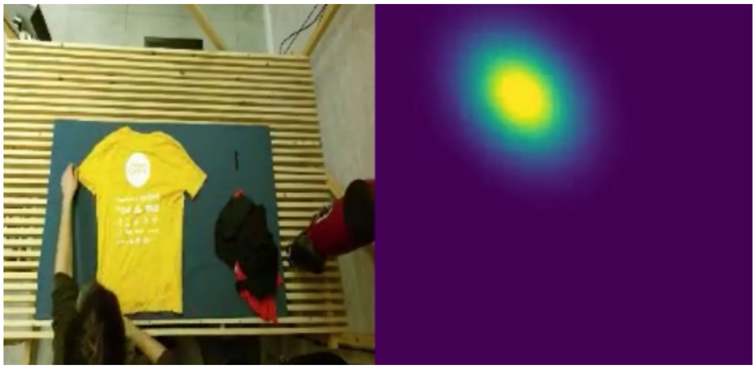
Loss functions are an important component for training neural networks as they determine the criteria for optimization. In this regard, future work can experiment and compare different existing loss functions like InfoNCE (van den Oord, Li, and Vinyals 2018) and cycle-consistency loss (Dwibedi et al. 2019). Furthermore, new contrastive loss functions can be researched in order to learn a more fine-grained representation in the last stages of training. This is of relevance for cloth folding in order to distinguish between folding results with and without wrinkles.

Similar to research in the contrastive loss function, architectural design choices influence the performance of the embedding. This is due to borrowing a backbone neural network from supervised learning and appending transform heads on top of it. However, choosing the appropriate transform head is non-trivial. For example, we found that using a multi-layered transform head performed better than a single non-linear readout layer. This observation suggests that a potential area of research is the architectural design and tradeoffs associated with representation pretraining and how it compares to supervised learning.

A final, future avenue we propose comes in the form of the representation. Currently, our embedding is specified as a vector of scalar values. However, throughout this research we have stressed the difficulty of cloth state estimation due to self-occlusions. This uncertainty is something humans deal intuitively with. For example, we can implicitly construct a heatmap of the most likely places a folded corner of a cloth to be at. Similarly, a probabilistic embedding can denote a range of values and uncertainty about the encoding of the inputs to embedding space. This way, the inherent uncertainty of the state estimation can be taken into account in the reward function. This can be done by appending, for example, Gaussian mixture layers on top of a base encoder network. The results discussed in Chapter 6, then become to interpret the probability of being in a certain space in the embedding. Figure 8.2 demonstrates preliminary results when following a similar training approach with a probabilistic embedding. The benefit of this approach is that it naturally encodes uncertainty when not all information is available. This uncertainty is present in our folding dataset: when demonstrators fold, for example, a sleeve and return their hands, it is expected that they move their hands to the other side of the shirt to fold the other sleeve. However, sometimes the demonstrators return their hands to the same sleeve to further refine the fold. Such uncertainty can be modelled by, for example, a Gaussian mixture model containing multiple distributions in the mixture to express the input image is possibly at two places in embedding space. In addition to the benefit of modelling uncertainty in a domain where uncertainty is inherently present, a probabilistic approach integrates well within the paradigm of distributional RL where an agent receives a distribution of return rather than the expectation of this return (Bellemare, Dabney, and Munos 2017). It has been shown that learning a value distribution outperforms maximizing an expected return, especially in the presence of the instabilities caused by function approximators such as deep neural networks.



(a) Deterministic embedding trained with the methodology of Chapter 6. Note that this image contains a temporal dimension encoded in the color of the points.



(b) Probablistic embedding trained with the methodology of Chapter 6. The colors represent the probability of the image being in a certain part of the embedding space. Compared to the image above, this image shows the encoding of one image and does not encode a temporal dimension.

Figure 8.2 Deterministic vs. probabilistic embedding. A probabilistic embedding space allows expressing how certain the network is about the state of the environment.

A

A

Cloth simulation and training parameters

A

Cloth simulation and training parameters

Simulation parameter tuning is necessary for the cloth simulation developed in Chapter 3 to be stable and resemble cloth-like behavior. In Table A.1, we give the used parameters we found to be stable and useful in our simulation setup. Table A.2 summarizes the parameters used for training the agent to learn to fold the simulated cloth. The weights of the reward functions components were tuned empirically by trial-and-error.

Table A.1 The used cloth simulation physical parameters.

Parameter	Value
Gravity multiplier	0.25
Cloth time subdivision	20
Integration scheme	Verlet
Elastic spring constant	5400
Shear spring constant	3000
Bending spring constant	2400
Inverse mass	0.125
Physics damping	38
Restitution constant*	0.025
Friction constant†	0.95
Elastic spring constant for meshes	2400
Shear spring constant for meshes	2400

* Remaining relative velocity after collision.
† Friction on surfaces due to collision.

Table A.2 The used training parameters to learn cloth folding in simulation.

Parameter	Value
Learning rate	0.001
State space size	$\mathbb{R}^{114}$
Action space size	$\mathbb{R}^{42}$
Reward function weight w_1	0.45
Reward function weight w_2	0.45
Reward function weight w_3	0.1
Neural network architecture	MLP
Nr hidden layers	2
Nr hidden neurons	128×64
Discount factor γ	0.95
Minibatch size k	1024
Target network update freq C	4096
Prioritization factor α	0.6
Importance sampling exponent β	0.4

Bibliography

Abbeel, Pieter, and Andrew Y Ng. 2004. "Apprenticeship learning via inverse reinforcement learning." In *Proceedings of the twenty-first international conference on Machine learning,* 1. ACM.

Akkaya, OpenAI: Ilge, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, et al. 2019. *Solving Rubik's cube with a robot hand.* arXiv: 1910.07 113 [cs.LG].

Alspach, Alex, Kunimatsu Hashimoto, Naveen Kuppuswamy, and Russ Tedrake. 2019. "Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation." In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft),* 597–604. https://doi.org/10.1109/ROBOSOFT.2019.8722713.

Andersson, R. L. 1987. "Real time expert system to control a robot ping-pong player." UMI Order No. GAX87-14001. PhD diss.

Andrychowicz, Marcin, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. "Hindsight experience replay." *arXiv preprint arXiv:1707.01495.*

Angelova, Anelia, Gustavo Carneiro, Kevin Murphy, Niko Sünderhauf, Jürgen Leitner, Ian Lenz, Trung T Pham, Vijay Kumar, Ingmar Posner, and Michael Milford. 2017. *Workshop on deep learning in robotic vision.*

Antonova, Rika, Peiyang Shi, Hang Yin, Zehang Weng, and Danica Kragic Jensfelt. 2021. "Dynamic environments with deformable objects."

Aomura, Shigeru, and Atsushi Koguchi. 2002. "Optimized bending sequences of sheet metal bending by robot." *Robotics and Computer-Integrated Manufacturing* 18 (1): 29–39.

Argall, Brenna D, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. "A survey of robot learning from demonstration." *Robotics and autonomous systems* 57 (5): 469–483.

Arriola-Rios, Veronica E., Puren Guler, Fanny Ficuciello, Danica Kragic, Bruno Siciliano, and Jeremy L. Wyatt. 2020. "Modeling of deformable objects for robotic manipulation: a tutorial and review." *Frontiers in Robotics and AI* 7:82. ISSN: 2296-9144. https://doi.org/10.3389/frobt.2020.00082. https://www.frontiersin.org/article/10.3389/frobt.2020.00082.

Baillargeon, Renée, Jie Li, Yael Gertner, and Di Wu. 2010. "How do infants reason about physical events" [in English (US)]. In *The Wiley-Blackwell Handbook of Childhood Cognitive Development, Second edition,* 11–48. United States: Wiley-Blackwell, July. ISBN: 9781405191166. https://doi.org/10.1002/9781444325485.ch1.

Balaguer, B., and S. Carpin. 2011. "Combining imitation and reinforcement learning to fold deformable planar objects." In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 1405–1412. September. https://doi.org/10.1109/IROS.2011.6094992.

Balakuntala, Mythra V., Upinder Kaur, Xin Ma, Juan Wachs, and Richard M. Voyles. 2021. *Learning multimodal contact-rich skills from demonstrations without reward engineering.* arXiv: 2103.01296 [cs.RO].

Balkcom, Devin J., and Matthew T. Mason. 2008. "Robotic origami folding." *The International Journal of Robotics Research* 27 (5): 613–627. https://doi.org/10.1177/0278364908090235. eprint: https://doi.org/10.1177/0278364908090235. https://doi.org/10.1177/0278364908090235.

Becht, Etienne, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. 2019. "Dimensionality reduction for visualizing single-cell data using UMAP." *Nature biotechnology* 37 (1): 38–44.

Bellemare, Marc G, Will Dabney, and Rémi Munos. 2017. "A distributional perspective on reinforcement learning." In *International Conference on Machine Learning,* 449–458. PMLR.

Bellman, Richard, and Robert Kalaba. 1959. "On adaptive control processes." *IRE Transactions on Automatic Control* 4 (2): 1–9.

Berg, Jur van den, Stephen Miller, Kenneth Goldberg, and Pieter Abbeel. 2010. "Gravity-based robotic cloth folding," 68:409–424. January. ISBN: 978-3-642-17451-3. https://doi.org/10.1007/978-3-642-17452-0_24.

Bersch, Christian, Benjamin Pitzer, and Sören Kammel. 2011. "Bimanual robotic cloth manipulation for laundry folding." In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 1413–1419. https://doi.org/10.1109/IROS.2011.6095109.

Bertiche, Hugo, Meysam Madadi, and Sergio Escalera. 2020. "CLOTH3D: clothed 3D humans." In *European Conference on Computer Vision,* 344–359. Springer.

Bicchi, Antonio. 1995. "On the closure properties of robotic grasping." *The International Journal of Robotics Research* 14 (4): 319–334. https://doi.org/10.1177/027836499501400402. eprint: https://doi.org/10.1177/027836499501400402. https://doi.org/10.1177/027836499501400402.

Billard, Aude, and Danica Kragic. 2019. "Trends and challenges in robot manipulation." *Science* 364 (6446): eaat8414. https://doi.org/10.1126/science.aat8414.

Bishop, Christopher M. 2006. "Pattern recognition." *Machine learning* 128 (9).

Blinn, James F. 1977. "Models of light reflection for computer synthesized pictures." In *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques,* 192–198. SIGGRAPH '77. San Jose, California: Association for Computing Machinery. ISBN: 9781450373555. https://doi.org/10.1145/563858.563893. https://doi.org/10.1145/563858.563893.

Bojarski, Mariusz, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. "End to end learning for self-driving cars." *arXiv preprint arXiv:1604.07316.*

Borràs, Júlia, Guillem Alenyà, and Carme Torras. 2020a. "A grasping-centered analysis for cloth manipulation." *IEEE Transactions on Robotics* 36 (3): 924–936. https://doi.org/10.1109/TRO.2020.2986921.

———. 2020b. *Encoding cloth manipulations using a graph of states and transitions.* arXiv: 2009.14681 [cs.RO].

Bousmalis, Konstantinos, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, et al. 2018. "Using simulation and domain adaptation to improve efficiency of deep robotic grasping." In *2018 IEEE international conference on robotics and automation (ICRA),* 4243–4250. IEEE.

Brass, Marcel, and Cecilia Heyes. 2005. "Imitation: is cognitive neuroscience solving the correspondence problem?" *Trends in cognitive sciences* 9 (November): 489–95. https://doi.org/10.1016/j.tics.2005.08.007.

Breiman, Leo. 2001. "Random forests." *Machine learning* 45 (1): 5–32.

Bro-Nielsen, M. 1998. "Finite element modeling in surgery simulation." *Proceedings of the IEEE* 86 (3): 490–503. https://doi.org/10.1109/5.662874.

Calandra, Roberto, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H. Adelson, and Sergey Levine. 2018. "More than a feeling: Learning to grasp and regrasp using vision and touch." *IEEE Robotics and Automation Letters* 3, no. 4 (October): 3300–3307. ISSN: 2377-3774. https : / / doi . org / 10 . 1109 / lra . 2018 . 2852779. http://dx.doi.org/10.1109/LRA.2018.2852779.

Cao, Junyue, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M Ibrahim, Andrew J Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J Steemers, et al. 2019. "The single-cell transcriptional landscape of mammalian organogenesis." *Nature* 566 (7745): 496–502.

Carter, Shan, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019. "Activation atlas." Https://distill.pub/2019/activation-atlas, *Distill,* https://doi.org/10.23915/distill.00015.

Case, Jennifer C, Joran Booth, Dylan S Shah, Michelle C Yuen, and Rebecca Kramer-Bottiglio. 2018. "State and stiffness estimation using robotic fabrics." In *2018 IEEE International Conference on Soft Robotics (RoboSoft),* 522–527. IEEE.

Case, Jennifer C., Michelle C. Yuen, Jane Jacobs, and Rebecca Kramer-Bottiglio. 2019. "Robotic skins that learn to control passive structures." *IEEE Robotics and Automation Letters* 4 (3): 2485–2492. https://doi.org/10.1109/LRA.2019.2906552.

Chebotar, Yevgen, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. 2019. *Closing the sim-to-real loop: adapting simulation randomization with real world experience.* arXiv: 1810.05687 [cs.RO].

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. "A simple framework for contrastive learning of visual representations." In *International conference on machine learning,* 1597–1607. PMLR.

Chi, Cheng, Xuguang Sun, Ning Xue, Tong Li, and Chang Liu. 2018. "Recent progress in technologies for tactile sensors." *Sensors* 18 (4). ISSN: 1424-8220. https://doi.org/10.3390/s18040948. https://www.mdpi.com/1424-8220/18/4/948.

Chua, Ping Yong, T Ilschner, and Darwin G Caldwell. 2003. "Robotic manipulation of food products–a review." *Industrial Robot: An International Journal.*

Chui, Haili, and Anand Rangarajan. 2003. "A new point matching algorithm for non-rigid registration." *Computer Vision and Image Understanding* 89 (2-3): 114–141.

Clocksin, W.F., J.S.E. Bromley, P.G. Davey, A.R. Vidler, and C.G. Morgan. 1985. "An implementation of model-based visual feedback for robot arc welding of thin sheet steel." *The International Journal of Robotics Research* 4 (1): 13–26. https://doi.org/10.1177/027836498500400102. eprint: https://doi.org/10.1177/027836498500400102. https://doi.org/10.1177/027836498500400102.

Cochrane, C., C. Hertleer, and A. Schwarz-Pfeiffer. 2016. "Smart textiles in health: An overview." In *Smart Textiles and their Applications,* edited by Vladan Koncar, 9–32. Woodhead Publishing Series in Textiles. Oxford: Woodhead Publishing. ISBN: 978-0-08-100574-3. https://doi.org/https://doi.org/10.1016/B978-0-08-100574-3.00002-3. https://www.sciencedirect.com/science/article/pii/B9780081005743000023.

Collins, Jack, Shelvin Chand, Anthony Vanderkop, and David Howard. 2021. "A review of physics simulators for robotic applications." *IEEE Access* 9:51416–51431. https://doi.org/10.1109/ACCESS.2021.3068769.

Corke, Peter I, et al. 1996. *Visual control of robots: high-performance visual servoing.* Research Studies Press Taunton, UK.

Coumans, Erwin. 2015. "Bullet physics simulation." In *ACM SIGGRAPH 2015 Courses.* SIGGRAPH '15. Los Angeles, California: Association for Computing Machinery. ISBN: 9781450336345. https://doi.org/10.1145/2776880.2792704. https://doi.org/10.1145/2776880.2792704.

Coumans, Erwin, and Yunfei Bai. 2016–2021. *PyBullet, a Python module for physics simulation for games, robotics and machine learning.* http://pybullet.org.

Cremers, Daniel. 2006. "Dynamical statistical shape priors for level set-based tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (8): 1262–1273.

Crooks, Whitney, Gabrielle Vukasin, Maeve O'Sullivan, William Messner, and Chris Rogers. 2016. "Fin ray® effect inspired soft robotic gripper: From the robosoft grand challenge toward optimization." *Frontiers in Robotics and AI* 3:70.

Cusumano-Towner, Marco, Arjun Singh, Stephen Miller, James F. O'Brien, and Pieter Abbeel. 2011. "Bringing clothing into desired configurations with limited perception." In *2011 IEEE International Conference on Robotics and Automation,* 3893–3900. https://doi.org/10.1109/ICRA.2011.5980327.

DARPA. 2015. "DRC final 2015." Accessed December 10, 2021. https://archive.darpa.mil/roboticschallenge/.

Dasari, Sudeep, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. 2019. "Robonet: large-scale multi-robot learning." *arXiv preprint arXiv:1910.11215.*

Degrave, Jonas, Michiel Hermans, Joni Dambre, and Francis wyffels. 2019. "A differentiable physics engine for deep learning in robotics." *Frontiers in Neurorobotics* 13:6. ISSN: 1662-5218. https://doi.org/10.3389/fnbot.2019.00006. https://www.frontiersin.org/article/10.3389/fnbot.2019.00006.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. "ImageNet: a large-scale hierarchical image database." In *CVPR09.*

Depierre, Amaury, Emmanuel Dellandréa, and Liming Chen. 2018. "Jacquard: a large scale dataset for robotic grasp detection." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 3511–3516. IEEE.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-training of deep bidirectional transformers for language understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers),* edited by Jill Burstein, Christy Doran, and Thamar Solorio, 4171–4186. Association for Computational Linguistics. https://doi.org/10.18653/v1/n19-1423. https://doi.org/10.18653/v1/n19-1423.

Diaz-Papkovich, Alex, Luke Anderson-Trocmé, Chief Ben-Eghan, and Simon Gravel. 2019. "UMAP reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts." *PLoS genetics* 15 (11): e1008432.

Dickmanns, Ernst Dieter, and Volker Graefe. 1988. "Applications of dynamic monocular machine vision." *Machine vision and Applications* 1 (4): 241–261.

Doersch, Carl, Abhinav Gupta, and Alexei A Efros. 2015. "Unsupervised visual representation learning by context prediction." In *Proceedings of the IEEE international conference on computer vision,* 1422–1430.

Donahue, Jeffrey, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. "Long-term recurrent convolutional networks for visual recognition and description." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 2625–2634.

Donaire, Sònia, Júlia Borràs, Guillem Alenyà, and Carme Torras. 2020. "A versatile gripper for cloth manipulation." *IEEE Robotics and Automation Letters* 5 (4): 6520–6527. https://doi.org/10.1109/LRA.2020.3015172.

Donlon, Elliott, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. 2018. "Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 1927–1934. IEEE.

Doumanoglou, Andreas, Andreas Kargakos, Tae-Kyun Kim, and Sotiris Malassiotis. 2014. “Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning.” In *2014 IEEE International Conference on Robotics and Automation (ICRA),* 987–993. https://doi.org/10.1109/ICRA.2014.6906974.

Doumanoglou, Andreas, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimir Petrik, Andreas Kargakos, Libor Wagner, Václav Hlaváč, Tae-Kyun Kim, and Sotiris Malassiotis. 2016. “Folding Clothes Autonomously: A Complete Pipeline.” *IEEE transactions on robotics* 32, no. 6 (December): 1461–1478. ISSN: 1552-3098. https://doi.org/10.1109/TRO.2016.2602376.

Drimus, Alin, Gert Kootstra, Arne Bilberg, and Danica Kragic. 2014. “Design of a flexible tactile sensor for classification of rigid and deformable objects.” *Robotics and Autonomous Systems* 62 (1): 3–15.

Droniou, Alain, Serena Ivaldi, and Olivier Sigaud. 2015. “Deep unsupervised network for multimodal perception, representation and classification.” Emerging Spatial Competences: From Machine Perception to Sensorimotor Intelligence, *Robotics and Autonomous Systems* 71:83–98. ISSN: 0921-8890. https://doi.org/https://doi.org/10.1016/j.robot.2014.11.005. https://www.sciencedirect.com/science/article/pii/S0921889014002474.

Duan, Yan, Marcin Andrychowicz, Bradly C Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. 2017. “One-shot imitation learning.” *arXiv preprint arXiv:1703.07326.*

Duflou, Joost R., József Váncza, and Richard Aerens. 2005. “Computer aided process planning for sheet metal bending: A state of the art.” *Computers in Industry* 56 (7): 747–771. ISSN: 0166-3615. https://doi.org/https://doi.org/10.1016/j.compind.2005.04.001. https://www.sciencedirect.com/science/article/pii/S0166361505000710.

Dwibedi, Debidatta, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2019. "Temporal cycle-consistency learning." In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June.

Dwibedi, Debidatta, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. 2018. "Learning actionable representations from visual observations." In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 1577–1584. IEEE. https://arxiv.org/abs/1808.00928.

Dzedzickis, Andrius, Ernestas Sutinys, Vytautas Bučinskas, Urte Bubniene, Baltramiejus Jakstys, Arunas Ramanavicius, and Inga Morkvenaite-Vilkonciene. 2020. "Polyethylene-carbon composite (Velostat®) based tactile sensor." *Polymers* 12 (December): 2905. https://doi.org/10.3390/polym12122905.

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

Elbrechter, Christof, Robert Haschke, and Helge Ritter. 2012. "Folding paper with anthropomorphic robot hands using real-time physics-based modeling." In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012),* 210–215. https://doi.org/10.1109/HUMANOIDS.2012.6651522.

English, P.B., M.J. Richardson, and C. Garzón-Galvis. 2018. "From crowdsourcing to extreme citizen science: participatory research for environmental health." PMID: 29608871, *Annual Review of Public Health* 39 (1): 335–350. https://doi.org/10.1146/annurev-publhealth-040617-013702. eprint: https://doi.org/10.1146/annurev-publhealth-040617-013702. https://doi.org/10.1146/annurev-publhealth-040617-013702.

Eppner, Clemens, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock. 2017. "Lessons from the Amazon picking challenge: four aspects of building robotic systems." In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17,* 4831–4835. https://doi.org/10.24963/ijcai.2017/676. https://doi.org/10.24963/ijcai.2017/676.

Erez, Tom, Yuval Tassa, and Emanuel Todorov. 2015. "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX." In *2015 IEEE International Conference on Robotics and Automation (ICRA),* 4397–4404. https://doi.org/10.1109/ICRA.2015.7139807.

Fang, Hao-Shu, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. 2017. "RMPE: regional multi-person pose estimation." In *ICCV.*

Fernando, Basura, Hakan Bilen, Efstratios Gavves, and Stephen Gould. 2017. "Self-supervised video representation learning with odd-one-out networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 3636–3645.

Finn, Chelsea, Sergey Levine, and Pieter Abbeel. 2016. "Guided cost learning: Deep inverse optimal control via policy optimization." In *International Conference on Machine Learning,* 49–58.

Finn, Chelsea, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. 2017. "One-shot visual imitation learning via meta-learning." In *Conference on Robot Learning,* 357–368. PMLR.

Fischler, Martin A, and Robert C Bolles. 1981. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM* 24 (6): 381–395.

Fleming, Peter. 2019. "Robots and organization studies: why robots might not want to steal your job." *Organization Studies* 40 (1): 23–38. https://doi.org/10.1177/0170840618765568. eprint: https://doi.org/10.1177/0170840618765568. https://doi.org/10.1177/0170840618765568.

Folgado, Duarte, Marília Barandas, Ricardo Matias, Rodrigo Martins, Miguel Carvalho, and Hugo Gamboa. 2018. "Time alignment measurement for time series." *Pattern Recognition* 81:268–279. ISSN: 0031-3203. https://doi.org/https://doi.org/10.1016/j.patcog.2018.04.003. http://www.sciencedirect.com/science/article/pii/S0031320318301286.

Foresti, G. L., and F. A. Pellegrino. 2004. "Automatic visual recognition of deformable objects for grasping and manipulation." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 34, no. 3 (August): 325–333. ISSN: 1094-6977. https://doi.org/10.1109/TSMCC.2003.819701.

Frank, Barbara, Rüdiger Schmedding, Cyrill Stachniss, Matthias Teschner, and Wolfram Burgard. 2010. "Learning the elasticity parameters of deformable objects with a manipulation robot." In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 1877–1883. https://doi.org/10.1109/IROS.2010.5653949.

Fu, Justin, Katie Luo, and Sergey Levine. 2018. "Learning robust rewards with adverserial inverse reinforcement learning." In *International Conference on Learning Representations.*

Gallese, Vittorio, Christian Keysers, and Giacomo Rizzolatti. 2004. "A unifying view of the basis of social cognition." *Trends in cognitive sciences* 8 (9): 396–403.

Gan, Chuang, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, et al. 2021. "ThreeDWorld: A platform for interactive multi-modal physical simulation." In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1).* https://openreview.net/forum?id=db1InWAwW2T.

Garcia-Camacho, Irene, Júlia Borràs, Berk Calli, Adam Norton, and Guillem Alenyà. 2021. *Household cloth object set: Fostering benchmarking in deformable object manipulation.* arXiv: 2111.01527 [cs.RO].

Garcia-Camacho, Irene, Martina Lippi, Michael C. Welle, Hang Yin, Rika Antonova, Anastasiia Varava, Julia Borras, et al. 2020. "Benchmarking bimanual cloth manipulation." *IEEE Robotics and Automation Letters* 5 (2): 1111–1118. https://doi.org/10.1109/LRA.2020.2965891.

Ge, Yuying, Ruimao Zhang, Xiaogang Wang, Xiaoou Tang, and Ping Luo. 2019. "DeepFashion2: a versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images." In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 5332–5340. https://doi.org/10.1109/CVPR.2019.00548.

Ge, Zhiqiang, Zhihuan Song, and Furong Gao. 2013. "Review of recent research on data-based process monitoring." *Industrial & Engineering Chemistry Research* 52 (10): 3543–3562. https://doi.org/10.1021/ie302069q. eprint: https://doi.org/10.1021/ie302069q. https://doi.org/10.1021/ie302069q.

Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2018. "Datasheets for datasets." *arXiv preprint arXiv:1803.09010.*

Giovanni, Stevie, and Kangkang Yin. 2011. "LocoTest: Deploying and evaluating physics-based locomotion on multiple simulation platforms," 227–241. November. ISBN: 978-3-642-25089-7. https://doi.org/10.1007/978-3-642-25090-3_20.

Giusti, Alessandro, Jérôme Guzzi, Dan C. Cireşan, Fang-Lin He, Juan P. Rodríguez, Flavio Fontana, Matthias Faessler, et al. 2016. "A machine learning approach to visual perception of forest trails for mobile robots." *IEEE Robotics and Automation Letters* 1 (2): 661–667. https://doi.org/10.1109/LRA.2015.2509024.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. 2011. "Deep sparse rectifier neural networks." In *Proceedings of the fourteenth international conference on artificial intelligence and statistics,* 315–323. JMLR Workshop and Conference Proceedings.

Golemo, Florian, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. 2018. "Sim-to-real transfer with neural-augmented robot simulation." In *Proceedings of The 2nd Conference on Robot Learning,* edited by Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, 87:817–828. Proceedings of Machine Learning Research. PMLR, 29–31 Oct. https://proceedings.mlr.press/v87/golemo18a.html.

Gomesh, N., I. Daut, V. Kumaran, M. Irwanto, Y.M. Irwan, and M. Fitra. 2013. "Photovoltaic powered t-shirt folding machine." TerraGreen 13 International Conference 2013 - Advancements in Renewable Energy and Clean Environment, *Energy Procedia* 36:313–322. ISSN: 1876-6102. https://doi.org/https://doi.org/10.1016/j.egypro.2013.07.036. https://www.sciencedirect.com/science/article/pii/S1876610213011223.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative adversarial nets." *Advances in neural information processing systems* 27.

Goodfellow, Ian J., Yoshua Bengio, and Aaron Courville. 2016. *Deep learning.* Http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press.

Graetz, Georg, and Guy Michaels. 2018. "Robots at work." *The Review of Economics and Statistics* 100, no. 5 (December): 753–768. ISSN: 0034-6535. https://doi.org/10.1162/rest_a_00754. eprint: https://direct.mit.edu/rest/article-pdf/100/5/753/1918863/rest_a_00754.pdf. https://doi.org/10.1162/rest%5C_a%5C_00754.

Graves, Alex, Greg Wayne, and Ivo Danihelka. 2014. *Neural turing machines.* arXiv: 1410.5401 [cs.NE].

Gu, Shixiang, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 2017. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates." In *2017 IEEE international conference on robotics and automation (ICRA),* 3389–3396. IEEE.

Guo, Yulan, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 2014. "3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey." *IEEE transactions on pattern analysis and machine intelligence* 36 (11): 2270–2287. https://doi.org/10.1109/TPAMI.2014.2316828.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. "Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor." In *International conference on machine learning,* 1861–1870. PMLR.

Hamajima, K., and M. Kakikura. 1996. "Planning strategy for unfolding task of clothes - isolation of clothes from washed mass." In *Proceedings of the 35th SICE Annual Conference. International Session Papers,* 1237–1242. https://doi.org/10.1109/SICE.1996.865443.

———. 1998. "Planning strategy for task untangling laundry - isolating clothes from a washed mass -." *J. Robotics Mechatronics* 10:244–251.

Harrell, RC, DC Slaughter, and Phillip D Adsit. 1989. "A fruit-tracking system for robotic harvesting." *Machine Vision and Applications* 2 (2): 69–80.

Hartikainen, Kristian, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. 2019. "Dynamical distance learning for semi-supervised and unsupervised skill discovery." In *International Conference on Learning Representations.*

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2001. *The elements of statistical learning.* Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Hausknecht, Matthew, and Peter Stone. 2016. "On-policy vs. off-policy updates for deep reinforcement learning." In *Deep Reinforcement Learning: Frontiers and Challenges, IJCAI 2016 Workshop.*

Havok. 2021. "Havok." Accessed October 10, 2021. https://www.havok.com/havok-physics/.

Heess, Nicolas, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, et al. 2017. *Emergence of locomotion behaviours in rich environments.* arXiv: 1707.02286 [cs.AI].

Heiden, Eric, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. 2021. "NeuralSim: Augmenting differentiable simulators with neural networks." In *2021 IEEE International Conference on Robotics and Automation (ICRA),* 9474–9481. IEEE.

Henderson, Peter, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. "Deep reinforcement learning that matters." In *Proceedings of the AAAI conference on artificial intelligence,* vol. 32. 1.

Hill, John. 1979. "Real time control of a robot with a mobile camera." In *9th Int. Symp. on Industrial Robots, 1979,* 233–246.

Hinton, Geoffrey E. 2002. "Training products of experts by minimizing contrastive divergence." *Neural computation* 14 (8): 1771–1800.

Ho, Jonathan, and Stefano Ermon. 2016. "Generative adversarial imitation learning." In *Advances in neural information processing systems,* 4565–4573.

Hoque, Ryan, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. 2020. "VisuoSpatial foresight for multi-step, multi-task fabric manipulation." In *Robotics: Science and Systems (RSS).*

Howard, Ayanna M, and George A Bekey. 2000. "Intelligent learning for deformable object manipulation." *Autonomous Robots* 9 (1): 51–58.

Hu, Yuanming, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. "Taichi: a language for high-performance computation on spatially sparse data structures." *ACM Transactions on Graphics (TOG)* 38 (6): 201.

Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. "Densely connected convolutional networks." In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2261–2269. https://doi.org/10.1109/CVPR.2017.243.

Huang, Isabella, Yashraj Narang, Clemens Eppner, Balakumar Sundaralingam, Miles Macklin, Tucker Hermans, and Dieter Fox. 2021. *DefGraspSim: Simulation-based grasping of 3D deformable objects.* arXiv: 2107.05778 [cs.RO].

Huang, Zhiao, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. 2021. "PlasticineLab: A soft-body manipulation benchmark with differentiable physics." *arXiv preprint arXiv:2104.03311.*

Hubel, David H, and Torsten N Wiesel. 1959. "Receptive fields of single neurones in the cat's striate cortex." *The Journal of physiology* 148 (3): 574–591.

Hutchinson, S., G.D. Hager, and P.I. Corke. 1996. "A tutorial on visual servo control." *IEEE transactions on robotics and automation* 12 (5): 651–670. https://doi.org/10.1109/70.538972.

Ibarz, Julian, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. 2021. "How to train your robot with deep reinforcement learning: Lessons we have learned." *The International Journal of Robotics Research* 40 (4-5): 698–721. https://doi.org/10.1177/0278364920987859. eprint: https : / / doi . org / 10 . 1177 / 0278364920987859. https://doi.org/10.1177/0278364920987859.

Imanberdiyev, Nursultan, Changhong Fu, Erdal Kayacan, and I-Ming Chen. 2016. "Autonomous navigation of UAV by using real-time model-based reinforcement learning." In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV),* 1–6. https://doi.org/10.1109/ICARCV.2016.7838739.

Inaba, Masayuki, and Hirochika Inoue. 1987. “Rope handling by a robot with visual feedback.” *Advanced Robotics* 2 (1): 39–54. https://doi.org/10.1163/156855387X00057. eprint: https://doi.org/10.1163/156855387X00057. https://doi.org/10.1163/156855387X00057.

Jakobsen, Thomas. 2001. “Advanced character physics.” *In Game Developers Conference Proceedings* (January).

James, Stephen, and Edward Johns. 2016. “3d simulation for robot arm control with deep q-learning.” *arXiv preprint arXiv:1609.03759.*

James, Stephen, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. 2019. “Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 12627–12637.

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

Janssens, Olivier, Rik Van de Walle, Mia Loccufier, and Sofie Van Hoecke. 2018. “Deep learning for infrared thermal image based machine health monitoring.” *IEEE/ASME Transactions on Mechatronics* 23 (1): 151–159. https://doi.org/10.1109/TMECH.2017.2722479.

Jia, Biao, Zhe Hu, Jia Pan, and Dinesh Manocha. 2018. “Manipulating highly deformable materials using a visual feedback dictionary.” In *2018 IEEE International Conference on Robotics and Automation (ICRA),* 239–246. https://doi.org/10.1109/ICRA.2018.8461264.

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

Jiménez, P. 2012. "Survey on model-based manipulation planning of deformable objects." *Robotics and computer-integrated manufacturing* 28 (2): 154–163.

Kang, Pilsung, Dongil Kim, and Sungzoon Cho. 2016. "Semi-supervised support vector regression based on self-training with label uncertainty: an application to virtual metrology in semiconductor manufacturing." *Expert Systems with Applications* 51:85–106. ISSN: 0957-4174. https://doi.org/https://doi.org/10.1016/j.eswa.2015.12.027. https://www.sciencedirect.com/science/article/pii/S0957417415008295.

Kappassov, Zhanat, Juan-Antonio Corrales, and Véronique Perdereau. 2015. "Tactile sensing in dexterous robot hands — Review." *Robotics and Autonomous Systems* 74:195–220. ISSN: 0921-8890. https://doi.org/https://doi.org/10.1016/j.robot.2015.07.015. https://www.sciencedirect.com/science/article/pii/S0921889015001621.

Kazerooni, H., and C. Foley. 2005. "A robotic mechanism for grasping sacks." *IEEE Transactions on automation science and engineering* 2 (2): 111–120. https://doi.org/10.1109/TASE.2005.844630.

Khalil, Fouad F, Pierre Payeur, and Ana-Maria Cretu. 2010. "Integrated multisensory robotic hand system for deformable object manipulation." In *Proc. Int. Conf. Robotics and Applications,* 159–166. Citeseer.

Kimura, Daiki, Ryutaro Nishimura, Akihiro Oguro, and Osamu Hasegawa. 2013. "Ultra-fast multimodal and online transfer learning on humanoid robots." In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction,* 165–166. HRI '13. Tokyo, Japan: IEEE Press. ISBN: 9781467330558.

Kingma, Diederik P, and Jimmy Ba. 2014. "Adam: a method for stochastic optimization." *arXiv preprint arXiv:1412.6980.*

Kirchheim, Alice, Matthias Burwinkel, and Wolfgang Echelmeyer. 2008. "Automatic unloading of heavy sacks from containers." In *2008 IEEE International Conference on Automation and Logistics,* 946–951. IEEE.

Kita, Y., and N. Kita. 2002. “A model-driven method of estimating the state of clothes for manipulating it.” In *Sixth IEEE Workshop on Applications of Computer Vision, 2002. (WACV 2002). Proceedings.* 63–69. https://doi.org/10.1109/ACV.2002.1182158.

Klingbeil, Ellen, Deepak Rao, Blake Carpenter, Varun Ganapathi, Andrew Y. Ng, and Oussama Khatib. 2011. “Grasping with application to an autonomous checkout robot.” In *2011 IEEE International Conference on Robotics and Automation,* 2837–2844. https://doi.org/10.1109/ICRA.2011.5980287.

Koenig, Nathan, and Andrew Howard. 2004. “Design and use paradigms for Gazebo, an open-source multi-robot simulator.” In *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2149–2154. Sendai, Japan, September.

Kohonen, Teuvo. 1982. “Self-organized formation of topologically correct feature maps.” *Biological cybernetics* 43 (1): 59–69.

Kristinsson, K., and G.A. Dumont. 1992. “System identification and control using genetic algorithms.” *IEEE Transactions on Systems, Man, and Cybernetics* 22 (5): 1033–1046. https://doi.org/10.1109/21.179842.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. “Imagenet classification with deep convolutional neural networks.” *Advances in neural information processing systems* 25:1097–1105.

Kroemer, Oliver, Scott Niekum, and George Konidaris. 2021. “A review of robot learning for manipulation: challenges, representations, and algorithms.” *J. Mach. Learn. Res.* 22:30–1.

Kuppuswamy, Naveen, Alex Alspach, Avinash Uttamchandani, Sam Creasey, Takuya Ikeda, and Russ Tedrake. 2020. “Soft-bubble grippers for robust and perceptive manipulation.” In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 9917–9924. https://doi.org/10.1109/IROS45743.2020.9341534.

Kuppuswamy, Naveen, Alejandro Castro, Calder Phillips-Grafflin, Alex Alspach, and Russ Tedrake. 2020. "Fast model-based contact patch and pose estimation for highly deformable dense-geometry tactile sensors." *IEEE Robotics and Automation Letters* 5 (2): 1811–1818. https://doi.org/10.1109/LRA.2019.2961050.

Kuzmanic, Ana, and Vlasta Zanchi. 2007. "Hand shape classification using DTW and LCSS as similarity measures for vision-based gesture recognition system." In *EUROCON 2007 - The International Conference on "Computer as a Tool",* 264–269. https://doi.org/10.1109/EURCON.2007.4400350.

Lambeta, Mike, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, et al. 2020. "DIGIT: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation." *IEEE Robotics and Automation Letters* 5, no. 3 (July): 3838–3845. ISSN: 2377-3774. https://doi.org/10.1109/lra.2020.2977257. http://dx.doi.org/10.1109/LRA.2020.2977257.

Laskin, Michael, Aravind Srinivas, and Pieter Abbeel. 2020. "CURL: Contrastive Unsupervised Representations for Reinforcement Learning." ArXiv:2003.06417, *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119.*

Laud, Adam Daniel. 2004. *Theory and application of reward shaping in reinforcement learning.* University of Illinois at Urbana-Champaign.

Lee, Hsin-Ying, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. 2017. "Unsupervised representation learning by sorting sequence." In *IEEE International Conference on Computer Vision.*

Lee, Michelle A, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. 2019. “Making sense of vision and touch: self-supervised learning of multimodal representations for contact-rich tasks.” In *2019 IEEE International Conference on Robotics and Automation (ICRA).* https://arxiv.org/abs/1810.10191.

Lee, Michelle A, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. 2020. “Making sense of vision and touch: Learning multimodal representations for contact-rich tasks.” *IEEE Transactions on Robotics* 36 (3): 582–596.

Lee, Robert, Daniel Ward, Akansel Cosgun, Vibhavari Dasagi, Peter Corke, and Jurgen Leitner. 2020. *Learning arbitrary-goal fabric folding with one hour of real robot experience.* arXiv: 2010.03209 [cs.RO].

Lei, Yaguo, Feng Jia, Jing Lin, Saibo Xing, and Steven X. Ding. 2016. “An Intelligent Fault Diagnosis Method Using Unsupervised Feature Learning Towards Mechanical Big Data.” *IEEE Transactions on industrial electronics* 63 (5): 3137–3147. https://doi.org/10.1109/TIE.2016.2519325.

Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. “End-to-end training of deep visuomotor policies.” *The Journal of Machine Learning Research* 17 (1): 1334–1373.

Levine, Sergey, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection.” *The International Journal of Robotics Research* 37 (4-5): 421–436. https://doi.org/10.1177/0278364917710318. eprint: https://doi.org/10.1177/0278364917710318. https://doi.org/10.1177/0278364917710318.

Leys, Christophe, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. 2013. "Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median." *Journal of Experimental Social Psychology* 49 (4): 764–766. ISSN: 0022-1031. https://doi.org/https://doi.org/10.1016/j.jesp.2013.03.013. http://www.sciencedirect.com/science/article/pii/S0022103113000668.

Li, Haisheng, and Xuefeng Zhu. 2004. "Application of support vector machine method in prediction of Kappa number of kraft pulping process." In *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788),* vol. 4, 3325–3330 Vol.4. https://doi.org/10.1109/WCICA.2004.1343151.

Li, Long, Tao Jin, Yingzhong Tian, Fei Yang, and Fengfeng Xi. 2019. "Design and Analysis of a Square-Shaped Continuum Robot With Better Grasping Ability." *IEEE Access* 7:57151–57162. https://doi.org/10.1109/ACCESS.2019.2914124.

Li, Yifei, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2021. *DiffCloth: Differentiable cloth simulation with dry frictional contact.* arXiv: 2106.05306 [cs.GR].

Li, Yinxiao, Chih-Fan Chen, and Peter K. Allen. 2014. "Recognition of deformable object category and pose." In *2014 IEEE International Conference on Robotics and Automation (ICRA),* 5558–5564. https://doi.org/10.1109/ICRA.2014.6907676.

Li, Yinxiao, Xiuhan Hu, Danfei Xu, Yonghao Yue, Eitan Grinspun, and Peter K Allen. 2016. "Multi-sensor surface analysis for robotic ironing." In *2016 IEEE International Conference on Robotics and Automation (ICRA),* 5670–5676. IEEE.

Li, Yinxiao, Yan Wang, Michael Case, Shih-Fu Chang, and Peter K Allen. 2014. "Real-time pose estimation of deformable objects using a volumetric approach." In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 1046–1052. IEEE.

Li, Yinxiao, Yonghao Yue, Danfei Xu, Eitan Grinspun, and Peter K. Allen. 2015. “Folding deformable objects using predictive simulation and trajectory optimization.” In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 6000–6006. https://doi.org/10.1109/IROS.2015.7354231.

Li, Yunzhu, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. 2018. “Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids.” *arXiv preprint arXiv:1810.01566.*

Li, Yunzhu, Jun-Yan Zhu, Russ Tedrake, and Antonio Torralba. 2019. “Connecting touch and vision via cross-modal prediction.” In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR).* June.

Liang, Junbang, Ming Lin, and Vladlen Koltun. 2019. “Differentiable cloth simulation for inverse problems.” In *Advances in Neural Information Processing Systems,* edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/28f0b864598a1291557bed248a998d4e-Paper.pdf.

Liang, Junbang, Ming C. Lin, and Vladlen Koltun. 2019. “Differentiable cloth simulation for inverse problems.” In *Conference on Neural Information Processing Systems (NeurIPS).*

Lillicrap, Timothy P, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. “Continuous control with deep reinforcement learning.” *arXiv preprint arXiv:1509.02971.*

Lin, Huan, Feng Guo, Feifei Wang, and Yan-Bin Jia. 2015. “Picking up a soft 3D object by “feeling” the grip.” *The International Journal of Robotics Research* 34 (11): 1361–1384. https://doi.org/10.1177/0278364914564232. eprint: https://doi.org/10.1177/0278364914564232. https://doi.org/10.1177/0278364914564232.

Lin, Long-Ji. 1992. *Reinforcement learning for robots using neural networks.* Carnegie Mellon University.

Lin, Xingyu, Yufei Wang, Jake Olkin, and David Held. 2020. "SoftGym: benchmarking deep reinforcement learning for deformable object manipulation." In *Conference on Robot Learning.*

Liu, Honghai, and Jian Dai. 2003. "An approach to carton-folding trajectory planning using dual robotic fingers." *Robotics and Autonomous Systems* 42 (1): 47–63.

Liu, Ziwei, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. 2016. "DeepFashion: powering robust clothes recognition and retrieval with rich annotations." In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 1096–1104. https://doi.org/10.1109/CVPR.2016.124.

Ljung, Lennart, and Torsten Söderström. 1983. *Theory and practice of recursive identification.* MIT press.

Lo Presti, Daniela, Chiara Romano, Carlo Massaroni, Jessica D'Abbraccio, Luca Massari, Michele Arturo Caponero, Calogero Maria Oddo, Domenico Formica, and Emiliano Schena. 2019. "Cardio-respiratory monitoring in archery using a smart textile based on flexible fiber bragg grating sensors." *Sensors* 19 (16). ISSN: 1424-8220. https://doi.org/10.3390/s19163581. https://www.mdpi.com/1424-8220/19/16/3581.

Lowe, David G. 1999. "Object recognition from local scale-invariant features." In *Proceedings of the seventh IEEE international conference on computer vision,* 2:1150–1157. Ieee.

Lu, Liang, and S. Akella. 1999. "Folding cartons with fixtures: a motion planning approach." In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C),* vol. 2, 1570–1576 vol.2. https://doi.org/10.1109/ROBOT.1999.772583.

Lyu, Yuting, Junghui Chen, and Zhihuan Song. 2019. "Image-based process monitoring using deep learning framework." *Chemometrics and intelligent laboratory systems* 189:8–17.

Machado, Inês P., A. Luísa Gomes, Hugo Gamboa, Vítor Paixão, and Rui M. Costa. 2015. "Human activity data discovery from triaxial accelerometer sensor: non-supervised learning sensitivity to feature extraction parametrization." *Information Processing & Management* 51 (2): 204–214. ISSN: 0306-4573. https://doi.org/https://doi.org/10.1016/j.ipm.2014.07.008. http://www.sciencedirect.com/science/article/pii/S0306457314000685.

MacKay-Lyons, Marilyn. 2002. "Central pattern generation of locomotion: A review of the evidence." *Physical Therapy* 82, no. 1 (January): 69–83. ISSN: 0031-9023. https://doi.org/10.1093/ptj/82.1.69. eprint: https://academic.oup.com/ptj/article-pdf/82/1/69/31663381/ptj0069.pdf. https://doi.org/10.1093/ptj/82.1.69.

Mahjourian, Reza, Risto Miikkulainen, Nevena Lazic, Sergey Levine, and Navdeep Jaitly. 2019. *Hierarchical policy design for sample-efficient learning of robot table tennis through self-play.* arXiv: 1811.12927 [cs.RO].

Mahler, Jeffrey, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. 2017a. "Dex-Net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics."

———. 2017b. "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics."

Mahler, Jeffrey, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. 2019. "Learning ambidextrous robot grasping policies." *Science Robotics* 4 (26): eaau4984.

Maitin-Shepard, J., M. Cusumano-Towner, J. Lei, and P. Abbeel. 2010. "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding." In *2010 IEEE International Conference on Robotics and Automation,* 2308–2315. May. https://doi.org/10.1109/ROBOT.2010.5509439.

Malhotra, Pankaj, Vishnu Tv, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. “Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder.” *1st SIGKDD Workshop on Machine Learning for Prognostics and Health Management* (August).

Mandlekar, Ajay, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. 2018. “Roboturk: a crowdsourcing platform for robotic skill learning through imitation.” In *Conference on Robot Learning,* 879–893. PMLR.

Matas, Jan, Stephen James, and Andrew J. Davison. 2018. “Sim-to-Real Reinforcement Learning for Deformable Object Manipulation.” In *Proceedings of the 2nd conference on robot learning,* edited by Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, 87:734–743. Proceedings of Machine Learning Research. PMLR, 29–31 Oct. http://proceedings.mlr.press/v87/matas18a.html.

McInnes, Leland, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. “UMAP: Uniform Manifold Approximation and Projection.” *Journal of Open Source Software* 3 (29): 861. https://doi.org/10.21105/joss.00861. https://doi.org/10.21105/joss.00861.

Mehrabi, Ninareh, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. “A survey on bias and fairness in machine learning.” *ACM Computing Surveys (CSUR)* 54 (6): 1–35.

Mehta, Bhairav, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. 2019. *Active domain randomization.* arXiv: 1904.04762 [cs.LG].

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. “Distributed representations of words and phrases and their compositionality.” In *Advances in neural information processing systems,* 3111–3119.

Miller, Stephen, Jur van den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. 2012. "A geometric approach to robotic laundry folding." *The International Journal of Robotics Research* 31 (2): 249–267. https://doi.org/10.1177/0278364911430417. eprint: https://doi.org/10.1177/0278364911430417. https://doi.org/10.1177/0278364911430417.

Misra, Ishan, C. Lawrence Zitnick, and Martial Hebert. 2016. "Shuffle and learn: unsupervised learning using temporal order verification." In *ECCV.*

Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. "Asynchronous methods for deep reinforcement learning." In *International conference on machine learning,* 1928–1937. PMLR.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. "Human-level control through deep reinforcement learning." *Nature* 518 (7540): 529.

Mochizuki, Jun, Michio Takahashi, and Seiji Hata. 1987. "Unpositioned workpieces handling robot with visual and force sensors." *IEEE Transactions on Industrial Electronics,* no. 1, 1–4.

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

Morita, T., J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi. 2003. "Knot planning from observation." In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422),* vol. 3, 3887–3892 vol.3. https://doi.org/10.1109/ROBOT.2003.1242193.

Morrison, D., A. W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, et al. 2018. *Cartman: The low-cost cartesian manipulator that won the Amazon robotics challenge.* arXiv: 1709.06283 [cs.RO].

Mrowca, Damian, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. 2018. "Flexible neural representation for physics prediction." *arXiv preprint arXiv:1806.08047.*

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

Müller, Matthias, Jos Stam, Doug James, and Nils Thürey. 2008. "Real time physics: class notes." In *ACM SIGGRAPH 2008 classes,* 1–90.

Murphy, Kevin P. 2012. *Machine learning: a probabilistic perspective.* 1st ed. Adaptive Computation and Machine Learning. The MIT Press. ISBN: 0262018020,9780262018029. http://gen.lib.rus.ec/book/index.php?md5=8ECFEEB2E1F9A19C770FBA1FF85FA566.

Myers, Cory, Lawrence Rabiner, and Aaron Rosenberg. 1980. "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28 (6): 623–635.

Nair, Ashvin, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. 2017. "Combining self-supervised learning and imitation for vision-based rope manipulation." In *2017 IEEE international conference on robotics and automation (ICRA),* 2146–2153. IEEE.

Nair, Ashvin V, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. 2018. "Visual reinforcement learning with imagined goals." In *Advances in Neural Information Processing Systems,* 9191–9200.

Nair, Suraj, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. 2020. "TRASS: time reversal as self-supervision." In *2020 IEEE International conference on Robotics and Automation (ICRA),* 115–121. https://doi.org/10.1109/ICRA40945.2020.9196862.

Narain, Rahul, Armin Samii, and James F O'brien. 2012. "Adaptive anisotropic remeshing for cloth simulation." *ACM transactions on graphics (TOG)* 31 (6): 1–10.

Navarro-Alarcon, David, Hiu Man Yip, Zerui Wang, Yun-Hui Liu, Fangxun Zhong, Tianxue Zhang, and Peng Li. 2016. "Automatic 3-D manipulation of soft objects by robotic arms with an adaptive deformation model." *IEEE Transactions on Robotics* 32 (March): 1–13. https://doi.org/10.1109/TRO.2016.2533639.

Nehaniv, Chrystopher L, Kerstin Dautenhahn, et al. 2002. "The correspondence problem." *Imitation in animals and artifacts* 41.

Ng, Andrew Y, H Jin Kim, Michael I Jordan, Shankar Sastry, and Shiv Ballianda. 2003. "Autonomous helicopter flight via reinforcement learning." In *NIPS,* vol. 16. Citeseer.

Ng, Andrew Y, Stuart J Russell, et al. 2000. "Algorithms for inverse reinforcement learning." In *Icml,* 1:2.

Nguyen, Anh, Jason Yosinski, and Jeff Clune. 2015. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 427–436.

Nguyen, Van-Duc. 1988. "Constructing force- closure grasps." *The International Journal of Robotics Research* 7 (3): 3–16. https://doi.org/10.1177/027836498800700301. eprint: https://doi.org/10.1177/027836498800700301. https://doi.org/10.1177/027836498800700301.

Niennattrakul, Vit, and Chotirat Ann Ratanamahatana. 2007. "On clustering multimedia time series data using K-means and dynamic time warping." In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07),* 733–738. https://doi.org/10.1109/MUE.2007.165.

Nvidia. 2021. "PhysX." Accessed October 10, 2021. https://developer.nvidia.com/gameworks-physx-overview.

Oh, Kyoung-Su, and Keechul Jung. 2004. "GPU implementation of neural networks." *Pattern Recognition* 37 (6): 1311–1314. ISSN: 0031-3203. https://doi.org/https://doi.org/10.1016/j.patcog.2004.01.013. https://www.sciencedirect.com/science/article/pii/S0031320304000524.

Ojha, Shipra, and Sachin Sakhare. 2015. "Image processing techniques for object tracking in video surveillance- A survey." In *2015 International Conference on Pervasive Computing (ICPC),* 1–6. https://doi.org/10.1109/PERVASIVE.2015.7087180.

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. "Curiosity-driven exploration by self-supervised prediction." In *International conference on machine learning,* 2778–2787. PMLR.

Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. "Context encoders: feature learning by inpainting." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 2536–2544.

Peng, Xue Bin, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. "Sim-to-real transfer of robotic control with dynamics randomization." *2018 IEEE International Conference on Robotics and Automation (ICRA)* (May). https://doi.org/10.1109/icra.2018.8460528. http://dx.doi.org/10.1109/ICRA.2018.8460528.

Peng, Xue Bin, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. 2020. "Learning agile robotic locomotion skills by imitating animals." *arXiv preprint arXiv:2004.00784.*

Petrík, Vladimír, Vladimír Smutný, Pavel Krsek, and Václav Hlaváč. 2017. "Single arm robotic garment folding path generation." *Advanced Robotics* 31 (23-24): 1325–1337. https://doi.org/10.1080/01691864.2017.1367325. eprint: https://doi.org/10.1080/01691864.2017.1367325. https://doi.org/10.1080/01691864.2017.1367325.

Pfaff, Tobias, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. 2021. "Learning mesh-based simulation with graph networks." In *International Conference on Learning Representations.* https://openreview.net/forum?id=roNqYL0_XP.

Pfeifer, Rolf. 2006. "Morphological computation – connecting brain, body, and environment." In *AI 2006: Advances in Artificial Intelligence,* edited by Abdul Sattar and Byeong-ho Kang, 3–4. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-49788-2.

Pham, Duc T, and Ashraf A Afify. 2005. "Machine-learning techniques and their applications in manufacturing." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 219 (5): 395–412.

Pinto, Lerrel, and Abhinav Gupta. 2016. "Supersizing self-supervision: learning to grasp from 50k tries and 700 robot hours." In *2016 IEEE international conference on robotics and automation (ICRA),* 3406–3413. IEEE.

Posner, Ingmar, Raia Hadsell, Martin Riedmiller, Markus Wulfmeier, and Rohan Paul. 2017. "Workshop on acting and interacting in the real world: Challenges in Robot Learning." In *Neural Information Processing Systems (NeurIPS).*

Proesmans, Remko, Andreas Verleysen, Robbe Vleugels, Paula Veske, Victor-Louis De Gusseme, and Francis Wyffels. 2022. "Modular piezoresistive smart textile for state estimation of cloths." *Sensors* 22 (1). ISSN: 1424-8220. https://doi.org/10.3390/s22010222. https://www.mdpi.com/1424-8220/22/1/222.

Provot, Xavier. 1995. "Deformation constraints in a mass-spring model to describe rigid cloth behavior."

Rabiner, Lawrence, and Biing-Hwang Juang. 1993. *Fundamentals of speech recognition.* USA: Prentice-Hall, Inc. ISBN: 0130151572.

Rajeswaran, Aravind, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations." *arXiv preprint arXiv:1709.10087.*

Ramachandram, Dhanesh, and Graham W Taylor. 2017. "Deep multimodal learning: A survey on recent advances and trends." *IEEE signal processing magazine* 34 (6): 96–108.

Ramisa, Arnau, Guillem Alenyà, Francesc Moreno-Noguer, and Carme Torras. 2012. "Using depth and appearance features for informed robot grasping of highly wrinkled clothes." In *2012 IEEE International Conference on Robotics and Automation,* 1703–1708. https://doi.org/10.1109/ICRA.2012.6225045.

———. 2013. "FINDDD: A fast 3D descriptor to characterize textiles for robot manipulation." In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 824–830. https://doi.org/10.1109/IROS.2013.6696446.

Rao, Kanishka, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. 2020. "Rl-cyclegan: Reinforcement learning aware simulation-to-real." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 11157–11166.

Raparthy, Sharath Chandra, Bhairav Mehta, Florian Golemo, and Liam Paull. 2020. "Generating automatic curricula via self-supervised active domain randomization." *arXiv preprint arXiv:2002.07911.*

Ravichandar, Harish, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. 2020. "Recent advances in robot learning from demonstration." *Annual Review of Control, Robotics, and Autonomous Systems* 3:297–330.

Redmon, Joseph, and Anelia Angelova. 2015. "Real-time grasp detection using convolutional neural networks." In *2015 IEEE International Conference on Robotics and Automation (ICRA),* 1316–1322. IEEE.

Remde, Axel, Dominik Henrich, and Heinz Wörn. 1999. "Picking-up deformable linear objects with industrial robots," http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-9550.

Riedmiller, Martin. 2005. "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method." In *European conference on machine learning,* 317–328. Springer.

Rizzi, Alfred A, and Daniel E Koditschek. 1993. "Preliminary experiments in spatial robot juggling." In *Experimental Robotics II,* 282–298. Springer.

Ross, Stéphane, Geoffrey Gordon, and Drew Bagnell. 2011. "A reduction of imitation learning and structured prediction to no-regret online learning." In *Proceedings of the fourteenth international conference on artificial intelligence and statistics,* 627–635.

Rubinstein, Reuven Y., and Dirk P. Kroese. 2004. *The cross entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation (information science and statistics).* Berlin, Heidelberg: Springer-Verlag. ISBN: 038721240X.

Rudin, Nikita, David Hoeller, Philipp Reist, and Marco Hutter. 2021. *Learning to walk in minutes using massively parallel deep reinforcement learning.* arXiv: 2109.11978 [cs.RO].

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams. 1986. "Learning representations by back-propagating errors." *nature* 323 (6088): 533–536.

Saadat, Mozafar, and Ping Nan. 2002. "Industrial applications of automatic manipulation of flexible materials." *Industrial Robot: An International Journal.*

Saha, Mitul, and Pekka Isto. 2007. "Manipulation planning for deformable linear objects." *IEEE Transactions on Robotics* 23 (6): 1141–1150. https://doi.org/10.1109/TRO.2007.907486.

Sajjan, Shreeyak S., Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. 2019. *ClearGrasp: 3D shape estimation of transparent objects for manipulation.* arXiv: 1910.02550 [cs.CV].

Salimans, Tim, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. "Evolution strategies as a scalable alternative to reinforcement learning." *arXiv preprint arXiv:1703.03864.*

Sallab, Ahmad EL, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. "Deep reinforcement learning framework for autonomous driving." *Electronic Imaging* 2017 (19): 70–76.

Samuel, Arthur L. 1959. "Some studies in machine learning using the game of checkers." *IBM Journal of research and development* 11 (6): 601–617.

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

Satish, Vishal, Jeffrey Mahler, and Ken Goldberg. 2019. "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks." *IEEE Robotics and Automation Letters* 4 (2): 1357–1364.

Saxena, Ashutosh, Justin Driemeyer, and Andrew Y Ng. 2008. "Robotic grasping of novel objects using vision." *The International Journal of Robotics Research* 27 (2): 157–173.

Schaul, Tom, John Quan, Ioannis Antonoglou, and David Silver. 2015. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952.*

Schneegass, Stefan, and Oliver Amft. 2017. *Smart textiles.* Springer.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin. 2015. "Facenet: A unified embedding for face recognition and clustering." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 815–823.

Schulman, John, Jonathan Ho, Cameron Lee, and Pieter Abbeel. 2016. "Learning from demonstrations through the use of non-rigid registration." In *Robotics Research,* 339–354. Springer.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. "Trust region policy optimization." In *International conference on machine learning,* 1889–1897. PMLR.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347.*

Seita, Daniel, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. 2021. *Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks.* arXiv: 2012.03385 [cs.RO].

Seita, Daniel, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, et al. 2020. "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Selfridge, Oliver, Richard Sutton, and Andrew Barto. 1985. "Training and tracking in robotics.," 670–672. January.

Sermanet, Pierre, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. 2018. "Time-contrastive networks: self-supervised learning from video." In *2018 IEEE International Conference on Robotics and Automation (ICRA),* 1134–1141. IEEE.

Seyler, Sean L, Avishek Kumar, Michael F Thorpe, and Oliver Beckstein. 2015. "Path similarity analysis: a method for quantifying macromolecular pathways." *PLoS Comput Biol* 11 (10): e1004568.

Shi, Haobin, Xuesi Li, Kao-Shing Hwang, Wei Pan, and Genjiu Xu. 2018. "Decoupled Visual Servoing With Fuzzy Q-Learning." *IEEE transactions on industrial informatics* 14 (1): 241–252. https://doi.org/10.1109/TII.2016.2617464.

Siciliano, Bruno, Oussama Khatib, and Torsten Kröger. 2008. *Springer handbook of robotics.* Vol. 200. Springer.

Silver, David, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, et al. 2016. "Mastering the game of go with deep neural networks and tree search." *Nature* 529 (January): 484–489. https://doi.org/10.1038/nature16961.

Singh, Avi, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. 2019. "End-to-end robotic reinforcement learning without reward engineering." *Robotics: Science and Systems.*

Smith, Russ. 2001. "Open dynamics engine." Accessed October 10, 2021. https://www.ode.org/.

Spectrum, I. 2015. "A compilation of robots falling down at the DARPA robotics challenge." Accessed December 10, 2021. www.youtube.com/watch?v=g0TaYhjpOfo&feature=youtu.be&t=26.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15 (1): 1929–1958.

Staranowicz, Aaron, and Gian Luca Mariottini. 2011. "A survey and comparison of commercial and open-source robotic simulator software." In *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments,* 1–8.

Stoppa, Matteo, and Alessandro Chiolerio. 2014. "Wearable electronics and smart textiles: a critical review." *sensors* 14 (7): 11957–11992.

Sukhbaatar, Sainbayar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. 2017. "Intrinsic motivation and automatic curricula via asymmetric self-play." *arXiv preprint arXiv:1703.05407.*

Sun, Chen, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. "Revisiting unreasonable effectiveness of data in deep learning era." In *Proceedings of the IEEE international conference on computer vision,* 843–852.

Sun, Li, Gerardo Aragon-Camarasa, Simon Rogers, and J. Paul Siebert. 2015. "Accurate garment surface analysis using an active stereo robot head with application to dual-arm flattening." In *2015 IEEE International Conference on Robotics and Automation (ICRA),* 185–192. https://doi.org/10.1109/ICRA.2015.7138998.

Sundaresan, Priya, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E Gonzalez, and Ken Goldberg. 2020. "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data." In *2020 IEEE International Conference on Robotics and Automation (ICRA),* 9411–9418. IEEE.

Sünderhauf, Niko, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, et al. 2018. "The limits and potentials of deep learning for robotics." *The International Journal of Robotics Research* 37 (4-5): 405–420. https://doi.org/10.1177/0278364918770733. eprint: https://doi.org/10.1177/0278364918770733. https://doi.org/10.1177/0278364918770733.

Sutton, Richard S, and Andrew G Barto. 2018. *Reinforcement learning: an introduction.* MIT press.

Sutton, Richard S, Andrew G Barto, and Ronald J Williams. 1992. "Reinforcement learning is direct adaptive optimal control." *IEEE Control Systems Magazine* 12 (2): 19–22.

Tagliabue, Eleonora, Ameya Pore, Diego Dall'Alba, Enrico Magnabosco, Marco Piccinelli, and Paolo Fiorini. 2020. "Soft tissue simulation environment to learn manipulation tasks in autonomous robotic surgery*." In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 3261–3266. https://doi.org/10.1109/IROS45743.2020.9341710.

Tan, Jie, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. 2018. *Sim-to-Real: Learning agile locomotion for quadruped robots.* arXiv: 1804.10332 [cs.RO].

Tanaka, D., S. Arnold, and K. Yamazaki. 2018. "EMD net: an encode–manipulate–decode network for cloth manipulation." *IEEE Robotics and Automation Letters* 3, no. 3 (July): 1771–1778. ISSN: 2377-3766. https://doi.org/10.1109/LRA.2018.2800122.

Taylor, Russell H., Arianna Menciassi, Gabor Fichtinger, Paolo Fiorini, and Paolo Dario. 2016. "Medical robotics and computer-integrated surgery." In *Springer Handbook of Robotics,* edited by Bruno Siciliano and Oussama Khatib, 1657–1684. Cham: Springer International Publishing. ISBN: 978-3-319-32552-1. https://doi.org/10.1007/978-3-319-32552-1_63. https://doi.org/10.1007/978-3-319-32552-1_63.

Tian, Stephen, Frederik Ebert, Dinesh Jayaraman, Mayur Mudigonda, Chelsea Finn, Roberto Calandra, and Sergey Levine. 2019. *Manipulation by feel: touch-based control with deep predictive models.* eprint: arXiv:1903.04128.

Tobin, Josh, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. "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),* 23–30. IEEE.

Todorov, Emanuel, Tom Erez, and Yuval Tassa. 2012. "MuJoCo: A physics engine for model-based control." In *IROS,* 5026–5033. IEEE. ISBN: 978-1-4673-1737-5. http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.

Tomo, Tito Pradhono, Massimo Regoli, Alexander Schmitz, Lorenzo Natale, Harris Kristanto, Sophon Somlor, Lorenzo Jamone, Giorgio Metta, and Shigeki Sugano. 2018. "A new silicone structure for uSkin—A soft, distributed, digital 3-Axis skin sensor and its integration on the humanoid robot iCub." *IEEE Robotics and Automation Letters* 3 (3): 2584–2591. https://doi.org/10.1109/LRA.2018.2812915.

Tormene, Paolo, Toni Giorgino, Silvana Quaglini, and Mario Stefanelli. 2009. "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation." *Artificial Intelligence in Medicine* 45 (1): 11–34. ISSN: 0933-3657. https://doi.org/https://doi.org/10.1016/j.artmed.2008.11.007. http://www.sciencedirect.com/science/article/pii/S0933365708001772.

Tsurumine, Yoshihisa, Yunduan Cui, Eiji Uchibe, and Takamitsu Matsubara. 2019. "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation." *Robotics and Autonomous Systems* 112:72–83. ISSN: 0921-8890. https://doi.org/https://doi.org/10.1016/j.robot.2018.11.004. https://www.sciencedirect.com/science/article/pii/S0921889018303245.

Twardon, Lukas, and Helge Ritter. 2015. "Interaction skills for a coat-check robot: Identifying and handling the boundary components of clothes." In *2015 IEEE International Conference on Robotics and Automation (ICRA),* 3682–3688. https://doi.org/10.1109/ICRA.2015.7139710.

Urbain, Gabriel, Victor Barasuol, Claudio Semini, J. Dambre, and Francis wyffels. 2021. "Effect of compliance on morphological control of dynamic locomotion with HyQ" [in eng]. *AUTONOMOUS ROBOTS* 45 (3): 421–434. ISSN: 0929-5593. %7Bhttp://dx.doi.org/10.1007/s10514-021-09974-9%7D.

van den Oord, Aäron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. "WaveNet: A generative model for raw audio." In *Arxiv.* https://arxiv.org/abs/1609.03499.

van den Oord, Aäron, Yazhe Li, and Oriol Vinyals. 2018. "Representation learning with contrastive predictive coding." *arXiv preprint arXiv:1807.03748.*

Van Hasselt, Hado, Arthur Guez, and David Silver. 2016. "Deep reinforcement learning with double q-learning." In *Proceedings of the AAAI conference on artificial intelligence,* vol. 30. 1.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention is all you need." In *Advances in neural information processing systems,* 5998–6008.

Vecerik, Mel, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2018. *Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards.* arXiv: 1707.08817 [cs.AI].

Verleysen, Andreas, Matthijs Biondina, and Francis wyffels. 2020. "Video dataset of human demonstrations of folding clothing for robotic folding." *The International Journal of Robotics Research,* ISSN: 0278-3649. http://dx.doi.org/10.1177/0278364920940408.

———. 2022. "Learning self-supervised task progression metrics: a case of cloth folding." *Applied Intelligence,* Accepted, https://doi.org/10.1007/s10489-022-03466-8.

Verleysen, Andreas, Thomas Holvoet, Remko Proesmans, Cedric Den Haese, and Francis wyffels. 2020. "Simpler learning of robotic manipulation of clothing by utilizing DIY smart textile technology." *APPLIED SCIENCES-BASEL* 10 (12): 10. ISSN: 2076-3417. http://dx.doi.org/10.3390/app10124088.

Verleysen, Andreas, and Francis wyffels. 2022 (expected). "Learning to fold cloth: a survey." *The International Journal of Robotics Research,* Submitted, under review.

Vinh, Trinh Van, Tetsuo Tomizawa, Shunsuke Kudoh, and Takashi Suehiro. 2012. "A new strategy for making a knot with a general-purpose arm." In *2012 IEEE International Conference on Robotics and Automation,* 2217–2222. https://doi.org/10.1109/ICRA.2012.6224852.

Wang, Ping Chuan, Stephen Miller, Mario Fritz, Trevor Darrell, and Pieter Abbeel. 2011. "Perception for the manipulation of socks." In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 4877–4884. https://doi.org/10.1109/IROS.2011.6095013.

Wang, Tuanfeng Y., Duygu Ceylan, Jovan Popović, and Niloy J. Mitra. 2018. “Learning a shared shape space for multimodal garment design.” *ACM Trans. Graph.* (New York, NY, USA) 37, no. 6 (December). ISSN: 0730-0301. https://doi.org/10.1145/3272127.3275074. https://doi.org/10.1145/3272127.3275074.

Wang, Xue Z. 2012. *Data mining and knowledge discovery for process monitoring and control.* Springer Science & Business Media.

Warneken, Felix, and Michael Tomasello. 2006. “Altruistic helping in human infants and young chimpanzees.” *science* 311 (5765): 1301–1303.

Watkins, Christopher JCH, and Peter Dayan. 1992. “Q-learning.” *Machine learning* 8 (3-4): 279–292.

Wen, Long, Xinyu Li, Liang Gao, and Yuyan Zhang. 2018. “A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method.” *IEEE Transactions on industrial electronics* 65 (7): 5990–5998. https://doi.org/10.1109/TIE.2017.2774777.

Willimon, B., S. Birchfield, and I. Walker. 2011. “Classification of clothing using interactive perception.” In *2011 IEEE International Conference on Robotics and Automation,* 1862–1868. May. https://doi.org/10.1109/ICRA.2011.5980336.

Willimon, Bryan, Stan Birchfield, and Ian Walker. 2011. “Model for unfolding laundry using interactive perception.” In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 4871–4876. https://doi.org/10.1109/IROS.2011.6095066.

Wu, Yilin, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. 2020. *Learning to manipulate deformable objects without demonstrations.* arXiv: 1910.13439 [cs.RO].

Wuest, Thorsten, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. 2016. “Machine learning in manufacturing: advantages, challenges, and applications.” *Production & Manufacturing Research* 4 (1): 23–45.

Xiang, Lingzhu, Florian Echtler, Christian Kerl, Thiemo Wiedemeyer, Lars, hanyazou, Ryan Gordon, et al. 2016. *libfreenect2: release 0.2,* April. https://doi.org/10.5281/zenodo.50641. https://doi.org/10.5281/zenodo.50641.

Xiao, Han, Kashif Rasul, and Roland Vollgraf. 2017. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.” *arXiv preprint arXiv:1708.07747.*

Yamaguchi, Akihiko, and Christopher G. Atkeson. 2017. “Implementing tactile behaviors using FingerVision.” In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids),* 241–248. https://doi.org/10.1109/HUMANOIDS.2017.8246881.

Yamakawa, Yuji, Akio Namiki, and Masatoshi Ishikawa. 2011. “Motion planning for dynamic folding of a cloth with two high-speed robot hands and two high-speed sliders.” In *2011 IEEE International Conference on Robotics and Automation,* 5486–5491. https://doi.org/10.1109/ICRA.2011.5979606.

Yamakawa, Yuji, Akio Namiki, Masatoshi Ishikawa, and Makoto Shimojo. 2008. “Knotting manipulation of a flexible rope by a multifingered hand system based on skill synthesis.” In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2691–2696. https://doi.org/10.1109/IROS.2008.4650802.

Yan, Mengyuan, Yilin Zhu, Ning Jin, and Jeannette Bohg. 2020. “Self-supervised learning of state estimation for manipulating deformable linear objects.” *IEEE Robotics and Automation Letters* 5 (2): 2372–2379. https://doi.org/10.1109/LRA.2020.2969931.

Yan, Wilson, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. 2020. “Learning predictive representations for deformable objects using contrastive estimation.” *CoRL.*

Yang, Pin-Chu, Kazuma Sasaki, Kanata Suzuki, Kei Kase, Shigeki Sugano, and Tetsuya Ogata. 2016. “Repeatable folding task by humanoid robot worker using deep learning.” *IEEE Robotics and Automation Letters* 2 (2): 397–403.

Yin, Shen, Steven X. Ding, Xiaochen Xie, and Hao Luo. 2014. “A review on basic data-driven approaches for industrial process monitoring.” *IEEE Transactions on Industrial Electronics* 61 (11): 6418–6428. https://doi.org/10.1109/TIE.2014.2301773.

You, Banseok, Chul Jong Han, Youngmin Kim, Byeong-Kwon Ju, and Jong-Woong Kim. 2016. "A wearable piezocapacitive pressure sensor with a single layer of silver nanowire-based elastomeric composite electrodes." *J. Mater. Chem. A* 4 (27): 10435–10443. https://doi.org/10.1039/C6TA02449A. http://dx.doi.org/10.1039/C6TA02449A.

Yuan, Wenzhen, Siyuan Dong, and Edward H. Adelson. 2017. "GelSight: high-resolution robot tactile sensors for estimating geometry and force." *Sensors* 17 (12). ISSN: 1424-8220. https://doi.org/10.3390/s17122762. https://www.mdpi.com/1424-8220/17/12/2762.

Yuan, Wenzhen, Yuchen Mo, Shaoxiong Wang, and Edward H Adelson. 2018. "Active clothing material perception using tactile sensing and deep learning." In *2018 IEEE International Conference on Robotics and Automation (ICRA),* 4842–4849. IEEE.

Yuen, Michelle C., Henry Tonoyan, Edward L. White, Maria Telleria, and Rebecca K. Kramer. 2017. "Fabric sensory sleeves for soft robot state estimation." In *2017 IEEE International Conference on Robotics and Automation (ICRA),* 5511–5518. https://doi.org/10.1109/ICRA.2017.7989649.

Zambelli, Martina, and Yiannis Demirisy. 2016. "Online multimodal ensemble learning using self-learned sensorimotor representations." *IEEE Transactions on Cognitive and Developmental Systems* 9 (2): 113–126.

Zhang, Fangyi, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. 2015. "Towards vision-based deep reinforcement learning for robotic motion control." *arXiv preprint arXiv:1511.03791.*

Zhang, Richard, Phillip Isola, and Alexei A Efros. 2016. "Colorful image colorization." In *European conference on computer vision,* 649–666. Springer.

Zhang, Tianhao, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. 2018. "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation." In *2018 IEEE International Conference on Robotics and Automation (ICRA),* 5628–5635. https://doi.org/10.1109/ICRA.2018.8461249.

Zhao, Rui, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X. Gao. 2019. "Deep learning and its applications to machine health monitoring." *Mechanical Systems and Signal Processing* 115:213–237. ISSN: 0888-3270. https://doi.org/https://doi.org/10.1016/j.ymssp.2018.05.050. https://www.sciencedirect.com/science/article/pii/S0888327018303108.

Zhao, Wenshuai, Jorge Peña Queralta, and Tomi Westerlund. 2020. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey." In *2020 IEEE Symposium Series on Computational Intelligence (SSCI),* 737–744. https://doi.org/10.1109/SSCI47803.2020.9308468.

Zhu, Heming, Yu Cao, Hang Jin, Weikai Chen, Dong Du, Zhangye Wang, Shuguang Cui, and Xiaoguang Han. 2020. "Deep Fashion3D: a dataset and benchmark for 3D garment reconstruction from single images." In *European Conference on Computer Vision,* 512–530. Springer.

Zhu, Yuke, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, et al. 2018. "Reinforcement and imitation learning for diverse visuomotor skills." In *Proceedings of Robotics: Science and Systems.* Pittsburgh, Pennsylvania, June. https://doi.org/10.15607/RSS.2018.XIV.009.

Zou, Liang, Chang Ge, Z. Jane Wang, Edmond Cretu, and Xiaoou Li. 2017. "Novel tactile sensor technology and smart tactile sensing systems: A review." *Sensors* 17 (11). ISSN: 1424-8220. https://doi.org/10.3390/s17112653. https://www.mdpi.com/1424-8220/17/11/2653.

Zou, Xingxing, Xiangheng Kong, Waikeung Wong, Congde Wang, Yuguang Liu, and Yang Cao. 2019. "FashionAI: a hierarchical dataset for fashion understanding." In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* 296–304. https://doi.org/10.1109/CVPRW.2019.00039.