Data-driven Robotic Manipulation of Cloth-like Deformable Objects: present, challenges, and future prospects

Halid Abdulrahim Kadi, Kasim Terzić

November 2022

Abstract

Manipulating cloth-like deformable objects (CDOs) is a long-standing problem in the robotics community. CDOs are flexible (non-rigid) objects that do not show a detectable level of compression strength while two points on the article are pushed towards each otherand include objects such as ropes (1D), fabrics (2D) and bags (3D). In general, CDOs' many degrees of freedom (DoF) introduce severe self-occlusion and complex state-action dynamics as significant obstacles for perception and manipulation systems. These challenges exacerbate existing issues of modern robotic control methods such as imitation learning (IL) and reinforcement learning (RL). This review focuses on the application details of data-driven control methods on four major task families in this domain: cloth-shaping, rope manipulation, dressing and bag manipulation. Furthermore, we identify specific inductive biases in these four domains that present challenges for more general IL and RL algorithms, and summarise the future direction for the development of the field.

1 Fundamentals of Robotics for CDO Manipulation

In this Section, following Kroemer et al. (2021) [199], we focus on providing the formalisation of a single centralised decision-making rigid-body agent. For specific details about existing CDO manipulation systems, see Section 4.

The minimum requirement for defining a robot is that it has to have actuators that it can control [200]. The objective of a robotic manipulation system (RMS) is to interact with its environment to change it to a specified goal configuration $\boldsymbol{g}$.

Earliest RMSs perform tasks with prescribed motion phases and analytical models to generate low-level control signals [262]. However, these types of systems are only practical under a closed and deterministic environment. In order to perform tasks in more stochastic environments, perception systems play a critical role in updating the robot's understanding of the configuration and changes in the environment [166, 134]. In classical robotics, a clear separation between perception and control is linked with intermediate representations. Perception is often handcrafted, as is the control procedure. The control system in robotics is often hierarchical, where the highest-level action abstraction is often prescribed by a heuristic, and lower-level control actions are delivered using motion planning and low-level controllers, such as PID and compliant controllers. The conversion between high-level action abstraction and low-level control signals is also often handcrafted in these systems. Details about classical robotics in CDO manipulation for individual domains will be discussed in Sections 4.1.3, 4.2.3, 4.3.3 and 4.4.3.

A dynamic model sometimes gets involved to provide prior estimation for the true state of the environment. It can also provide faster future roll-outs for planning and trajectory optimisation methods. The dynamical model often comes with the form of an approximated analytical model [273, 396]. Those are the early examples of optimal control methods, also known as Type I model-based reinforcement learning (MBRL) (Section 3.3), in classical robotics [360, 321].

Robotic applications often show a hierarchical structure on the action output. A long-horizon multi-step task can be decomposed into a sequence of subtasks. Also, some fundamental skills (Section 1.3) are used repeatedly across the subtasks. These skills can be further divided into multiple goal-condition phases that can be delivered by motion planning algorithms and low-level analytical controllers [199]. The modular hierarchical structure of a task decomposes the challenge into simpler and more tractable problems so that the agent can use the skills as the base action abstraction to perform a complex task. Such hierarchical action structure often requires corresponding hierarchical structure at the state representation [199].

1.1 Data-driven Control

Building analytical model for producing control signals is a difficult process, especially for CDOs due to their complex deformation behaviours. Controllers developed under classical robotics typically only apply to a fixed or a narrow range of configurations [166, 134]. One of the key features of modern robotics methods is that they formulate the manipulation problem as a *Partially Observable Markov Decision Process* (Section 1.2) and leverage contextual MDPs (Section 1.2) for developing more robust and general skill controllers (Section 1.3).

With the advance of deep learning [118] and high-performance hardware, almost all parts of an RMS can be practically replaced with neural networks (NNs) and trained on collected data. We refer to this evolution as the beginning of modern robotics because NNs fundamentally improved the capability of all parts of the system and revolutionised the system design process [162]. However, there is still a clear separation between perception and control; the selection of state representation is still an important step in the system design. NNs are also convenient and effective at combining multi-modal sensory inputs [84], as well as integrating passive and interactive perception [110, 223, 351]. In this era, IL using real-world demonstration data has become one of the most popular and relatively robust control algorithms in research and industry [278, 63]. Early examples of IL methods, such as behaviour cloning (BC) (Section 2.1) and learning from observation (LfO) (Section 2), with tabular or simple parametric policy representation show good performance in knot-tying tasks [259].

Meanwhile, the speed and quality of simulations has improved tremendously, making it possible to collect millions of data points using simulation for training the perception and control systems, and to train a non-linear parametric dynamic model. The high-performance simulation also improves the precision of planning/trajectory optimisation methods (Type I model-based reinforcement learning, Section 3.3) by replacing the analytical model with either the simulation or the learned dynamic model. Most importantly, it makes RL practical in robotic applications because simulation provides safe and fast online exploration and data collection. Consequently, *simulation-to-reality (Sim2Real)* transfer [419] (Section 1.4) becomes a key technology in modern robotics for bridging the gap between the simulation-trained policy and the real-world deployment.

End-to-end policy learning attempts to eliminate the selection of intermediate state representation from the design process by learning a latent representation so that a control algorithm can be applied to several domains. However, the NN parameters must be trained from scratch for different tasks. Hence, transfer learning is proposed to transfer the domain knowledge learned from a certain task to other domains, where it can be applied directly (zero-shot transfer) or be functional after a few trials (few-shot). Transfer learning is the key technology used in *Sim2Real* (Section 1.4) and Multi-task RL (Section 3.4). End-to-end policy learning suffers from data efficiency and mode collapse (similar but different state leads to the same action), so representation learning (Section 3.2) becomes essential [340]. It aims to learn better latent state estimation to overcome both issues, leading to robust and interpretable control systems (Section 1.5).

DRL and DIL often come with a set of hyper-parameters which have to be manually tuned separately for each domain. In order to automate this process, meta-learning attempts to dynamically adjust the hyper-parameters regarding its experience in the target domain and training status [291]. Another hand-designed aspect of many current robotics algorithms is the reward function. Adversarial inverse reinforcement learning (Adversarial IRL) methods (Section 5.2) also attempt to automate reward engineering with demonstration data while preserving similar computational complexity as their RL counterparts [150, 293].

Sections 4.1.4, 4.2.4, 4.3.4 and 4.4.3, will go into detail about current applications of data-driven approaches in the CDO domain.

1.2 Framework

Markov decision process (MDP) is a discrete-time stochastic control process first introduced by Bellman (1957) [30] and further developed by Howard (1960) [157]. It is a mathematical framework that models the decision-making process in a stochastic system and produces corresponding rewards for the decision steps. MDPs can be used to solve optimisation problems through dynamic programming, an instance of reinforcement learning algorithms. They have become a strong mathematical tool in many disciplines, including robotics. An MDP $\mathcal{M}$ is defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0)$, where

- $\mathcal{S}$ is a set of states,
- $\mathcal{A}$ is a set of actions,
- $\mathcal{P}^{\boldsymbol{a}}_{\boldsymbol{s},\boldsymbol{s}'} = \Pr(\boldsymbol{s}_{t+1} = \boldsymbol{s}' | s_t = s, \boldsymbol{a}_t = \boldsymbol{a})$ is the transition probability function,
- $\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s},\boldsymbol{s}'}$ is a primary reward function that produces a reward when transitioning from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ by taking action $\boldsymbol{a}$, and
- ρ_0 is the initial state distribution.

In robotics and reinforcement learning settings (Section 3), a task can be modelled by an MDP. The defining feature of an MDP is that it has the Markov property, i.e., the transition function and the reward function only depend on the current state and action not history trajectories. This means that while modelling a robotic application, we do not have to consider the effect of state-action sequences; we assume each state is unique and informative enough for transition and decision-making. When different tasks share the same transitional property but different reward functions, we say that they belong to the same dynamic-invariant task family. We are also interested in action-invariant task families, where an agent can solve different MDPs with a fixed action space.

Hidden Markov Model (HMM) [292] is a first-order Markov process whose internal states $\mathcal{S}$ are not directly observable. The observable output signal $\mathcal{X}$ depends on probability distributions $\mathcal{O}$. Since the model can disregard noise through a stochastic framework, it allows us to deal with the highly stochastic underlying structure of the process. HMMs can be fit using *Expectation-Maximisation (EM)* algorithms [71]. HMM is a tool to formalise trajectory behaviour cloning methods (Section 2.1.2) that has been used in dressing domain (Section 4.3.3). HMMs can also be leveraged to detect execution states of RMSs, such as success and various error cases, which is applied in dressing domain (Section 4.3.2). Also, they are commonly used to model perception systems [181, 68] that can provides prior to the state estimation for a manipulation task.

A **Partially Observable MDP [177]** $\mathcal{PM}$ combines a MDP and an HMM to model an agent's decision process where the agent can only partially observe the state of the domain. Other than the elements of MDP, a POMDP also contains $(\mathcal{X}, \mathcal{O})$ that models the observation space $\mathcal{X}$ and emission probability $\mathcal{O}$ of the observation $\boldsymbol{x}$ from a state $\boldsymbol{s}$. POMDPs more accurately describe the real-world application of robotic systems [162]. They are a tool for modern DRL and DIL algorithms that train end-to-end policy controllers.

Contextual MDP (CMDP) [132] is a tuple $\mathcal{CM} = (\Xi, \mathcal{M})$, where Ξ is the context space and $\mathcal{M}$ is a function that maps a context to a specific MDP. Hence, an MDP controlled by

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2022 doi:10.20944/preprints202212.0305.v1

this variable is $\mathcal{M}(\boldsymbol{\xi}) = \big(\mathcal{S}, \mathcal{A}, \mathcal{P}(\boldsymbol{\xi}), \mathcal{R}(\boldsymbol{\xi}), \rho_0(\boldsymbol{\xi})\big)$. It is possible to reformulate a CMDP as a big MDP by joining all possible elements and corresponding transitions. However, knowing the context variable $\boldsymbol{\xi}$ can make the decision-making problem easier than solving the compounded big MDP. Suppose the context does not affect the state-action space or the transition and reward function, it might be easier to integrate the variation of the MDP caused by the context into the initial state distribution ρ_0.

In robotics, context captures the across-task variations [199], and it can help to formalise robust and multi-task control settings. In fact, we can regard **Goal-condition MDP** as a special subset of CMDP, where a goal $\boldsymbol{g} \in \mathcal{G}$ only influences the reward function. Skill controllers (Section 1.3) and multi-task/goal-condition RL (Section 3.4) are built upon this framework, and the application of it will be discoursed in cloth-shaping tasks (Section 4.1.4.

In addition, we can also have **Contextual POMDP** (CPOMDP) that is defined as $\mathcal{CPM} = (\Xi, \mathcal{PM})$, where the observation space and emission distribution also depend on the context, i.e., $\big(\mathcal{X}(\boldsymbol{\xi}), \mathcal{O}(\boldsymbol{\xi})\big)$. CPOMDP is a framework for building robust end-to-end controller that can generalise across different contextual observational inputs. *Sim2Real* (Section 1.4) techniques such domain randomisation and domain adaptation adopt this framework to transfer the simulation-trained end-to-end policy to real world. The application of such methods to the cloth-shaping domain will be detailed Section 4.1.4.

A **Task Family** is a distribution on CMDPs, i.e., $P(\mathcal{CM})$, where each corresponds to a specific task. Usually, they share a similar (not exactly the same) dynamic and reward functions, but vary in context. These tasks are close enough to each other that the agent is expected to generalise across them rather than solve them separately with individual policies [199].

1.3 Skills

Skills are a high-level action abstraction can be reapplied throughout different phases of a task and even across task families. For example, laundry folding requires applying cloth-grasping, cloth-flattening and cloth-flattening skills multiple times to perform the long-horizon multi-step task successfully. Some skills require part-based representations, i.e. if the agent detects a handle on a bag, it knows where to grasp the object [74, 200]. Other skills relate to the mode switch in the environment, where the actuation of the environment changes in a piecewise fashion [199].

In robotics, skills are often modelled using the **Option** framework [354]. Each motor skill can be represented as an option $\omega = (I_\omega, \beta_\omega, \pi_\omega)$, where $I_\omega : \mathcal{S} \to \{0, 1\}$ is an indicator function that allows the option to execute; $\beta_\omega : \mathcal{S} \to [0, 1]$ is a termination probability on the current option, and π_ω is the policy function for the option. Apart from state representation, some skill policies are also subject to the context of the environment. Given a skill library Ω, i.e., a collection of skills, an agent is expected to know when to execute and terminate a certain skill ω, meaning that the agent needs to know the pre-condition and post-condition of executing a certain skill.

In addition to having a firm knowledge of the pre-condition and post-condition of a skill, the skill controller π_ω should also be flexible and robust to a range of initial states $\mathcal{S}_\omega$, context Ξ_ω and even goal conditions $\mathcal{G}_\omega$. This requires a contextual goal-condition skill policy represented as:

$$\pi_\omega : \mathcal{S}_\omega[\times \Xi_\omega[\times \mathcal{G}_\omega]] \to \mathcal{A}_\omega | \mathcal{T}^{A_\omega} \quad , \tag{1}$$

where the output of the skill policy can be a single action $\boldsymbol{a} \in \mathcal{A}_\omega$ or a trajectory of actions $\tau \in \mathcal{T}^{A_\omega}$, while the input of the policy differs depending on the application.

Solving long-horizon multi-step problems usually involves (1) Decomposing a task into component skills w.r.t. the modular structure, (2) developing skill policies independently, and (3) using higher level policy to decide when to use which skills. A skill library offers a procedural abstraction to the agent, meaning that it decides to execute an option based on the state abstraction. This supports abstract task-level planning and improves the interpretability of the system [200]. There are two major approaches for integrating skills: (1) segmenting demonstration trajectories into component skills or (2) manually including skill specification as part of the overall problem when learning to solve a task [199], which is mainly used in laundry folding (Section 4.1.3) and knot-tying/untying (Section 4.2.3). Autonomous skill segmentation and discovery [90] could be an interesting direction to explore in CDO manipulation.

Furthermore, transferring the learned skill to another task domain helps the learning efficiency and generalisation ability of the agent. We can initialise a new skill policy with a learned skill policy and fine-tune it in the new task domain [281], which requires a consistent state representation between the two task domains. Transfer learning techniques (Section 3.4) such as domain adaptation and domain randomisation [365] are often utilised to build perception systems that produce unified state abstraction for different domains. The same technology is used to transfer the simulation-trained policy to the real world (Section 1.4).

1.4 Simulation to Reality

Simulation-based approaches provide cheap and safe development environment for RMSes, but there are challenges to deploying simulation-trained systems into real-world settings [419]. The major challenge is domain shift: the simulated MDP differs from the real-world MDP. There is a considerable difference in the observation space and some mismatches in the transitional functions. Also, there is always a risk that the agent will encounter novel states in the actual trials that it does not encounter in simulation [295].

In classical robotics, the mismatch caused by the dynamic model is partially mitigated through system identification [198] that calibrates the simulator with real-world data. The perception mismatch is resolved with a fixed intermediate state representation between the simulation and the reality. However, since the intermediate state representation is automatically learned in an end-to-end manner in modern robotics, small perturbations in the observation space can cause a significant difference in the latent state representation; consequently, they can lead to a substantial deviation from the correct policy.

In DL, transfer learning (TL) aims at improving the performance of a learner on a target domain by transferring the knowledge from a different but related source domain [420]. **Domain adaptation** is the main TL method that maps the observations emitted by the same state from the two domains into the same state abstraction. **Domain randomisation** [261] is another technique used to fill the observation space gap. While domain adaptation attempts to unify the two domains, domain randomisation feeds the agent with an observation from a set of the randomised domains.

Apart from bridging the gap in the observation space, we also want to transfer the policy from the simulation to reality. In the ideal case, with a robust perception system that gives a consistent intermediate representation and perfectly tuned simulation dynamic, the control system will not have trouble deploying its policy from simulation to reality. This is called zero-shot or direct transfer. The quality of the physics engines has been improving drastically, and more realistic robotic simulators are being developed to fill the gap between the transitional functions in the real world and simulation [366, 332, 107].

In addition, we cannot guarantee perfect simulation in practice. As a result, a simulation-trained agent often runs into novel states in reality. One solution is to adopt continual learning [367] to let the agent fine-tune its policy with the new data collected in the physical trials while preserving its knowledge in the simulation. One such technique is called policy distillation [307], where a smaller student network is trained with the data generated by a pre-trained teacher network. Meta RL [381] is another solution that aims to train an agent that can learn a new policy faster in a new domain using the knowledge from past trainings. Alternatively, Robust RL [258] attempts to resolve the policy gap by adding perturbation in the dynamic of the simulation, which shares some commonalities with domain randomisation. Furthermore, a real demonstration trajectory can be provided to let the agent adjust its policy using imitation learning methods.

1.5 Safety

Safety is another major concern in robotic applications, especially in the real world. A robot arm needs to avoid jiggering motion and reaching toward regions outside the boundary of its working space to prevent damaging its own body, sensors and motors. It also needs to avoid collisions and exerting excessive forces on objects, humans and collaborators. For example, the robot needs to avoid exerting large forces on the user in assistive dressing (Section 4.3) and tearing CDO in bimanual manipulations.

With access to a sufficient state representation, these requirements can be met using smoothed motion planning and low-level compliant controllers [43]. Nevertheless, exploration of RL algorithms brings challenges for training the robot safely in the real world. Although one can use simulators to train the agent in simulation and deploy it to the real world, the robot will typically require extra exploration to adjust to the physical trials. Safe RL [113] becomes an important topic for modern robotic applications. One solution for Safe RL is adding risk metrics to the reward function and constraints to the action space to penalise the agent from utilising destructive actions and reaching unsafe states [60, 155]. Another solution is initialising the policy with a demonstration trajectory and exploring around the trajectory to fine-tune its policy [358] (Section 3.5.2). Section 4.3.3 and 4.3.4 will discuss the application of conventional safe control strategies and Safe RL in the dressing domain.

2 Imitation Learning

Imitation learning (IL) [272] is the first type of data-drive control methods that has been used considerably in all four families of CDO domain (Section 4). Also known as learning from demonstration (LfD) [315], IL learns policy from demonstration as the raw input:

$$\mathrm{IL}(\pi^{demo}) \doteq \arg\min_{\pi\in\Pi} D\big[p(\pi^{demo})||p(\pi)\big] \quad , \tag{2}$$

where D is the divergence metric that measures the difference between the two distributions; the operands in the divergence can be state-action distribution, trajectory feature expectation or trajectory feature distribution. Most of the IL algorithm is related to M-projection, i.e., choosing KL divergence from the demonstration distribution to the target policy distribution, while I-projection-related algorithms have barely been investigated in the IL literature [272]. The fundamental difference between IL and RL is that no reward functions are provided in IL. IL helps to eliminate the effort of engineering reward that can be difficult in many robotics applications.

Behaviour cloning (BC) (Section 2.1) is a type of IL that attempts to directly learn the policy from the demonstration in a way similar to supervised learning. Inverse reinforcement learning (IRL) [306] (Section 5), or inverse optimal control [179], is the second type of IL, which learns the reward function from the demonstration policy and learns the control policy from the inferred reward function. IL can also vary in terms of the control signal given in the demonstration data. Most common IL method applied in CDO is BC, and we are aware of no applications of IRL in the domain.

Learning from Observation (LfO), or Imitation from Observation, refers to imitating learning approaches while the action signals are not in the demonstration data. It learns only from state/observation demonstration. LfO is often applied when the demonstrator's action space differs from the agent's. A direct way to achieve LfO is to provide the learner with the keyframes, a sequence of intermediate observations/states, the uses lower-level controllers to achieve these goals. For example, Morita et al. (2003) [259] initiate *Knot Planning form Observation (KPO)* that integrates control methods from LfO on the topological level of the rope, where the changes of the representation in the consecutive demonstrated observation produce a primitive action. Human imitation learning attempts to infer the intent of the demonstration and probably make a different action than the demonstration. This is analogous to doing LfO using IRL, where the reward function only on state/observation signals [69].

Even though imitation learning sounds like an ideal approach to tackle robotic applications and, in fact, BC is the most prevalent method in practice [272], it still suffers from several issues. The major bottlenecks are from the demonstration data that can be (1) suboptimal, (2) contain noise and (3) have a different domain (different MDP/POMDP) compared to the application domain.

Demonstration data can be collected through (1) kinesthetic teaching, (2) a motion capture system, (3) teleoperation system and (4) an expert script policy. Kinesthetic teaching is leveraged frequently in assistive dressing systems, where a human coordinator grabs the robot arm to demonstrate dressing trajectories, and the data are collected trough the robot's sensorimotors [358]. Motion capture systems collect the demonstration data by keeping track of the movement of a physical expert performing a task. However, such data often present a correspondence problem between the demonstrator and the learner. Teleoperation, on the other hand, requires a human to operate a controller to drive the robot to perform a task. For example, Seita et al. (2019) [329] demonstrate pick-and-place action strategies in vision space to perform bed-making through a click interface. Vinh et al. (2012) [373] employ a joystick to control the gripper's location and orientation to demonstrate single-gripper overhand knot-tying in the air. Finally, expert-script policy are functional systems that can perform the target task relatively good in real world, or a privileged policy that can get access to the true state of the environment in simulation. The advantage of such a data-collection system is that it frees human demonstration. Such demonstrators have seen substantial use in cloth-shaping systems [329, 234, 155] as well as knot-tying systems [356].

As with reinforcement learning, the dynamic model can improve the data efficiency of IL approaches. In model-free imitation learning (MFIL), the dynamic of the environment is implicitly encoded into the learned policy. However, these models often need more samples to converge to a smooth and stable optimal solution. In many RMSs, kinematic controllers are well-developed for control joints. Such systems are fully actuated, and MFIL methods can be easily applied in such applications. In contrast, adopting MFIL algorithms in under-actuated systems is challenging due to the compounding error it may encounter. Model-based imitation learning (MBIL) attempts to learn a policy that regenerates the demonstrated trajectories by learning/using the dynamic model of the under-actuated system [272]. MFIL is mainly used for trajectory learning problems, while MBIL is mainly applied to object manipulation tasks.

2.1 Behaviour Cloning

Behaviour Cloning (BC) [21] achieves IL by learning the policy directly from demonstration data. Concerning policy abstraction, BC can be classified into methods that learn state-action policy, trajectory-level planning, and task-level planning. Regarding the involvement of a dynamic model, BC can also be classed into model-free and model-based approaches [272]. Model-based BC (MBBC) [22, 263] methods are mainly adopted to solve the **corresponding problem** [37]. A corresponding problem often appears when the embodiment of the demonstrator differs from the learner. Furthermore, it is hard to apply trajectory-level MFBC to underactuated systems where the set of reachable states is limited. In contrast, it is possible to plan a feasible trajectory close to the demonstration trajectory in such settings using a dynamic model [12]. In this subsection, we only focus on the details of MFBC algorithms.

A general algorithmic framework of MFBC is:(1) Collect demonstration data $\mathcal{D}$. (2) Select a policy representation $\hat{\pi}$. (3) Select an objective function $\mathcal{L}$. (4) Optimise $\mathcal{L}$ w.r.t. $\hat{\pi}$. using $\mathcal{D}$. A simple approach for BC is to use purely supervised learning [289], but it cannot learn to recover from failures during the test time because of the **compounding error**, i.e., a cascade of errors from the learned policy. In a supervised approach to BC, we assume that actions in the expert trajectories are independent and identically distributed. Nevertheless, causal relationships exist in BC applications between the action and the states, which contradicts the assumption of the independent and identical distribution of the data while using SL. Pure SL methods cannot capture the structure of the task domain. Additionally, SL methods cannot capture other constraints of the robot system, such as joint limits and workspace configuration [20]. Sections 4.1.4, 4.2.4, 4.3.4 and 4.4.3 will discuss the application of SL methods in CDO manipulation.

2.1.1 State-action BC

Ross and Bagnell (2010) [302] propose *Forward Training* to mitigate the compounding error using time-dependent policies. It iteratively collects expert policies on each time step that ends with the sampled trajectories induced by previously learned policies and trains the current step policy using the current-step demonstrations. However, it requires running the dynamic for sampling trajectories throughout the iteration, and the task horizon can be larger, even undefined. These make the algorithm impracticable in real-world trials. There are other stochastic mixing methods, such as *Search-based structured prediction (SEARN)* [70] and *Stochastic Mixing Iterative Learning (SMILe)* [302], that attempt to replace the expert policy gradually throughout the optimisations. Chernova and Veloso (2009) [56] also propose a confidence-based method to gather extra demonstration data while the confidence is lower than a threshold. In order to tackle the suboptimality problem of demonstration data, Kim et al. (2013) [184] propose *Approximate Policy Iteration with Demonstration (APID)* that is a BC method regularised with *Approximate Policy Iteration.*

Ross et al. (2011) [304] propose *Dataset Aggregation (DAgger)* to solve compounding error of BC that can be regarded as a reduction of imitation learning to supervised learning with interaction. Unlike the methods mentioned above, *DAgger* maintains one learning policy and iteratively optimises it by collecting aggregated demonstration data. He et al. (2012) [139] propose *DAgger by Coaching* to mitigate the learning difficulty of *DAgger* due to the gap between the learning policy and demonstration policy. They do so by replacing the demonstration policy with a hope action policy that is easier to learn than the demonstration policy. Venkatraman et al. (2015) [375] propose a framework called *Data as Demonstrator* (DaD) to extend the idea of *DAgger* to multi-step prediction. Section 4.1.4 will talk about the application of DAgger in cloth-shaping tasks.

As an extension of *DAgger*, Ross and Bagnell (2014) [303] present *AggreVaTe* to learn a policy that maximises the reward-to-go of the demonstration policy instead of using supervised BC loss. This is formally defined as follows

$$\mathcal{L}_{AggreVaTe}(\pi, i) = -\frac{1}{T}\sum_{t=1}^{T} \mathop{\mathbb{E}}_{\boldsymbol{s}_t \sim d_t^{\pi_i}} \left[\mathop{\mathbb{E}}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s}_t)} \left[R_{T-t}^{\pi^{\text{demo}}}(\boldsymbol{s}_t, \boldsymbol{a})) \right] \right] \quad , \tag{3}$$

where reward-to-go $R_t^{\pi^{\text{demo}}}(\boldsymbol{s}, \boldsymbol{a})$ is assumed to be known or can be estimated without bias after rolling out using mixed policy $\pi_i = \beta_i \pi^{\text{demo}} + (1-\beta_i)\hat{\pi}_i$ for $t-1$ steps and π^{demo} for $T-t$ steps. *AggreVaTe* can be interpreted as a regret reduction of imitation learning to no-regret online learning. Using the reward-to-go function, this approach can overcome the suboptimality problem of the demonstration data. Sun et al. (2017) [347] propose *Deep AggreVaTe (AggreVaTeD)* that uses a deep neural network to learn more expressive policies than *AggreVaTe.*

In robotics, Zeng et al. (2021) [411] propose *Transporter Net* to solve vision-based pick-and-place BC using NNs. *Transporter Net* leverages two convolutional networks $Q_{pick}(\boldsymbol{x})$ and $Q_{place}(\cdot|\boldsymbol{x}, \boldsymbol{a}_{pick})$, where the latter is composed of key and query networks. They output state-action value map for picking and placing position in pixel space. The architecture learns to detect the pick action $\boldsymbol{a}_{pick} = \arg\max_{x,y} Q_{pick}((x, y)|\boldsymbol{x})$, and uses the feature map of the region-of-interest of the pick action $\boldsymbol{x}[\boldsymbol{a}_{pick}]$ to cross-correlate to the feature map of the observation $\boldsymbol{x}$

for estimating its place action, i.e., $\boldsymbol{a}_{place} = \arg\max_{x,y} f_{query}(\boldsymbol{x}[\boldsymbol{a}_{pick}]) * f_{key}(\boldsymbol{x})$. The whole inference process is trained end-to-end using demonstration data. Seita et al. (2021) [327] extend *Transporter Net* to goal-condition control domain and apply it on cloth-shaping and bag-manipulation tasks in simulation. This is discussed in detail in Section 4.1.4 and Section 4.4.3.

2.1.2 Trajectory BC

The simplest form of Trajectory BC is trajectory replay, where the system detects a state-of-interest and then replays the associated demonstrated trajectory. However, these methods are only functional under a small state subset of the task domain, and they often require extra control to reach and detect these states-of-interest. Vinh et al. (2012) [373] and Kudoh et al. (2015) [202] apply such methods to achieve knot-tying tasks, which will be addressed in Section 4.2.4.

One direct way to model trajectory learning is using HMM. However, an HMM cannot produce long smooth action sequences as it is a discrete process [272]. Calinon et al. (2010) [45] leverage Gaussian models with HMM to represent continuous values. Yu (2010) [407] proposes *Hidden Semi-Markov Model* (HSMM) to formulate state duration distribution, and Rozo et al. (2016) [305] adopts *Linear Quadratic Regulator (LQR)* [31] to optimise the trajectory generated by an HSMM. The application of such framework in assistive dressing will be presented in Section 4.3.4.

Dynamic Movement Primitives (DMP) [163, 165, 164, 316] is a trajectory behaviour cloning method that formalises trajectory learning problem as a damped forced harmonic oscillator system. It produces smooth movement to reach the goal within the desired duration:

$$\gamma^2 \ddot{\boldsymbol{s}} = \alpha_{\boldsymbol{s}}\big(\beta_{\boldsymbol{s}}(\boldsymbol{g} - \boldsymbol{s}) - \gamma \dot{\boldsymbol{s}}\big) + f \quad , \tag{4}$$

where $(\alpha_{\boldsymbol{s}}, \beta_{\boldsymbol{s}})$ are constants that control damping and spring factors individually; γ is a constant that control temporal behaviour; and the goal configuration $\boldsymbol{g}$ is given by the last state of a demonstration trajectory. The force controller is represented by a weighted combination of basis functions $\{\psi_i\}_{i=1}^N$, usually Gaussian functions. These weights are learned by minimising the sum square error between demonstrated target force and the controller force across time. Section 4.3.4 will discuss about the application of *DMP* in assistive dressing.

There are many variants in the literature of *DMPs* for different purposes. Hoffmann et al. (2009) [152] add a term in the actuator system to achieve obstacle avoidance, while Deniša et al. (2016) [72] propose *Compliant Movement Primitives* for compliant motions. Mulling et al. (2013) [260] propose *Mixture of motor primitives (MoMP)* to combine multiple *DMPs* for generating a more robust trajectory controller. Moreover, due to their deterministic nature, *DMPs* cannot simulate human trajectory behaviour. Paraschos et al. (2013) [279] propose *Probabilistic Movement Primitives (ProMPs)* that bring stochasticity into the controller by representing the controlling behaviour as a distribution over demonstrated trajectories. However, ProMPs does not guarantee the stability of the motion trajectory. The above-mentioned trajectory BC method is time-dependent. Khansari-Zadeh and Billard (2011) [183] propose *Stable Estimator of Dynamical Systems (SEDS)* that represents time-invariant dynamical systems with *Gaussian Mixture Model* (GMM). Although *SEDS* cannot learn time-dependent behaviour by nature, it produces a stable trajectory to reach the target configuration.

DMP works well for learning point-to-point trajectories as it can easily generalise to dierent start and goal positions. However, *DMP* generally has limited capability for generalisation and often requires more features for different usage. Gaussian distributions are used in *PoMPs* and *SEDs* for better generalisation behaviour. Schulman et al. (2016) [324], on the other hand, propose to transfer the demonstration trajectory using non-rigid registration between the observed scene and a demonstrated scene. They conduct the registration using *Thin Plate Spline Robust Point Matching (TPS-RPM)* approach proposed by Chui and Rangarajan (2003)

[57] on the scene represented as point clouds. The application of TPS-RPM has mainly been explored in rope-manipulation tasks, which will be discussed in Section 4.2.4. Lee et al. (2015a) [218] extend the work to force control, while Lee et al. (2015b) [216] replace *TPS-RPM* with *Coherent Point Drift* to improve the transferring quality. These approaches eliminate the effort of modelling the distribution over demonstrated trajectories like in *PoDMP* and *SEDS*.

The learned controllers can produce poor trajectories when encountering a new situation if the given demonstration trajectories are suboptimal and limited. Incremental trajectory learning methods can improve the learner's behaviour by providing corrective actions [88, 240]. In addition, the mixing strategy of different demonstration trajectories is also a key to improving the robust behaviour of the controller [260, 88].

2.1.3 Task-level BC

Most of the trajectory-level BC can only be applied to learn kinematic and compliant control in robotics so that they can be used to learn primitive actions and skills. A higher-level controller can generate a sequence of such primitive actions and skills to perform a complex task. This is known as task-level planning. In task-level BC, it is expected to learn to perform long-horizon multi-step tasks automatically from demonstration trajectories. This usually requires segmenting and clustering the provided trajectories automatically.

Konidaris et al. (2012) [196] propose to learn hierarchical structure for skills from demonstration. The tree-like structure is formed by merging similar skill chains, produced with change point detection [96]. Manschitz et al. (2015) [243] propose to learn the transitional function among various action primitives using *Support Vector Machine (SVM)*. Kroemer et al. (2014) [201] propose *Autoregressive Hidden Markov Model* to represent a long-horizon task as a sequence of *DMPs*. By letting the latent variable condition the observable variable, the transition of MDPs can be executed based on the observations. Niekum et al. (2014) [268] suggest to segment the trajectory with *Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM)* [104] and training the task-level control with a finite-state machine.

3 Reinforcement Learning for CDO manipulation

Reinforcement learning is the second type of data-driven approach used in CDO manipulation systems. The goal of reinforcement learning (RL) [352], or optimal control [35], is to optimise the policy π by maximising the expected future accumulative reward, known as return R [36].

$$\mathrm{RL}(\mathcal{M}) \doteq \arg\max_{\pi\in\Pi} \mathop{\mathbb{E}}_{\tau\sim d^{\pi}}\left[\sum_{t=1}^{T} r_t\right], \text{or} \mathop{\mathbb{E}}_{\tau\sim d^{\pi}}\left[\sum_{t=1}^{T}\gamma^{t-1} r_t\right], \tag{5}$$

where τ is a trajectory from the distribution d^{π} of trajectories induced by the MDP and policy π; r_t represents the reward collected at step t, which is equivalent to $\mathcal{R}^{\boldsymbol{a}_{t-1}}_{\boldsymbol{s}_{t-1},\boldsymbol{s}_t}$; and γ is the discount factor between 0 and 1 to control the importance of a reward with the increment of its collected future step, and it is often adopted in the infinite-horizon case to bound the trajectory return. In the case of deterministic dynamic or stochastic dynamic with low uncertainty, we can approximate $r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ with $r(\boldsymbol{s}, \boldsymbol{a})$.

RL can be classified into model-free reinforcement learning (MFRL) (Section 3.1) and model-based reinforcement learning (MBRL) (Section 3.3) w.r.t. the learned/existing dynamic model. In the following subsections, we will talk about the clear difference between these two algorithm families. RL can also be categorised in terms of the way of using training data: (1) off-policy RL agent trains on the data collected using different policies than the current RL policy, but the agent is still allowed to explore the environment; (2) offline RL agent, a special case of an off-policy agent, trains only on pre-collected data; and (3) on-policy RL agent, which is only allowed to update its policy based on the trajectories generated by its current policy. Additionally, maximum entropy RL (MaxEnt-RL) (Section 3.5.1) incorporate the *Maximum Causal Entropy Principle* to model human's stochastic behaviour and, in turn, produce better exploration policy and robust target policy. Contemporary RL algorithms, usually DRL algorithms, vary in terms of their representation learning (Section 3.2) and exploration strategies (Section 3.5).

Exploration and **reward shaping** have been the two significant challenges to apply reinforcement learning in practice. With the introduction of deep NNs, reinforcement learning algorithms have become practical in high-dimensional, even continuous state settings. However, this introduces many other new challenges [162]: (1) theoretically, we have **no guarantee of convergence** of the DRL algorithms; (2) **data-efficiency** (both sampling and learning efficiency) is a major concern for such complex settings; (3) DRL algorithms often come with a set of **hyper-parameters**, and the algorithms are often sensitive to these hyper-parameters and application domain. (4) High-dimensional/continuous state settings also exacerbate the challenge of **exploration** and, in turn, worsen the data-efficiency issue. Complex **dynamic** of high dimensional/continuous state-action settings further aggravate the above-mentioned challenges.

The application of DRL in robotics also helps to overcome the challenge of state estimation in POMDP settings [352]. Similarly, it automatise the perception in robotics through end-to-end training and combining multi-modal sensory inputs in the perception network [199]. However, it also further introduces new challenges and amplifies the existing challenges in DRL. First, DRL is data-hungry and trained through trial-and-error which requires the **human involvement** for constantly resetting the environment and preventing the robot from unsafe actions [162]. Second, **reward function** needs to be engineered from perception, and this defeats the idea of tackling the perception in an unsupervised manner. Also, even with an oracle (where the agent has access to the true state), the reward design is a hard problem for robotics. Third, robotics is usually formulated in a **partially observable** setting, where estimating the true state of the environment is a hard problem. Fourth, RL always deals with **unknown dynamics** of the environment. Last but not least, **asynchronous control**, i.e., the delay between action and the sensory input, violates the formulation of MDP, especially for dynamic control settings.

Use of DRL also opens up more possibilities in robotics, like creating multi-task robots (Section 3.4). Conventionally, robots are only designed to perform a single task with slight variations in context. The generalisability of NNs holds promise of an agent that can perform different types of tasks across different MDPs. However, the **specification of goals** and **inference of the reward function** for such an agent remains a hard challenge.

There are many available benchmark environments for the development of DRL. *Atari 2600* computer games are the most popular benchmark environments for testing discrete action RL agents in POMDP settings. The *Arcade Learning Environment (ALE)* [29] implemented 55 Atari games for the usage of testing intelligent agents. *DeepMind-Lab* [26] is a first-person benchmark environment that is often used to test the exploration ability of an RL algorithm in long-horizon complex tasks. This benchmark is framed in POMDP setting with low-dimension discrete action space. *Open-AI Gym* [40] contains a wide range of MDP/POMDP environments with both continuous and discrete actions, including Atari games, board games and robotic control environments. This benchmark also offers the classic and toy environments for tabular RL settings. Built up on *Mujoco*, benchmark environment `dm_control` suits is proposed for developing and evaluating RL agents on continuous MDP/POMDP settings [359]. This benchmark includes a collection of environments with several settings that aim to learn different skill types, and some even have the option to learn with sparse rewards.

3.1 Model-free RL

Model-free Reinforcement Learning (MFRL) aims to solve the sequential decision making problem without the knowledge of dynamics. It is mainly classified into value-based methods and policy-gradient methods — the final control policy is either choosing the action that leads to the best state-action value (Section 3.1.1) or directly optimising policy against the RL objective

(Section 3.1.2). To scale these two classes of methods into continuous action settings, actor-critic (AC) methods are derived to assist the learning in practice. AC methods learn parameterised value and policy functions at the same time. Note that some AC methods derive from value-based methods, and some from policy-gradient methods, so we will not discuss AC algorithms separately.

3.1.1 Value-based Methods

Discrete action value-based DRL *Q-learning* has been applied in cloth-shaping (Section 4.1.4) and knot-untying (Section 4.2.4). Continuous action methods, such as *DDPG* and *SAC*, have mainly been adopted in cloth-shaping tasks (Section 4.1.4). For understanding these methods, this section will introduce the evolution of value-base methods from *Q-learning* to *DDPG*.

Value-based methods choose the action that maximises the expected future return at each environment step. Here, we define state $V^\pi(\boldsymbol{s})$ and state-action $Q^\pi(\boldsymbol{s},\boldsymbol{a})$ **value functions** of an MDP (Section 1.2) and their self-consistent **Bellman equations** [352]:

$$V^\pi(\boldsymbol{s}) = \mathop{\mathbb{E}}_{\tau\sim d^\pi}\big[R(\tau) \mid \boldsymbol{s}_1 = \boldsymbol{s}\big] = \mathop{\mathbb{E}}_{\boldsymbol{a}\sim\pi(\cdot|\boldsymbol{s}),\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},\boldsymbol{a})}\big[\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s},\boldsymbol{s}'} + \gamma\, V^\pi(\boldsymbol{s}')\big] \tag{6}$$

$$Q^\pi(\boldsymbol{s},\boldsymbol{a}) = \mathop{\mathbb{E}}_{\tau\sim d^\pi}\big[R(\tau) \mid \boldsymbol{s}_1 = \boldsymbol{s}, \boldsymbol{a}_1 = \boldsymbol{a}\big] = \mathop{\mathbb{E}}_{\boldsymbol{s}'\sim\mathcal{P}(\cdot|\boldsymbol{s},\boldsymbol{a})}\big[\mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s},\boldsymbol{s}'} + \gamma \mathop{\mathbb{E}}_{\boldsymbol{a}'\sim\pi(\cdot|\boldsymbol{s}')}\big[Q^\pi(\boldsymbol{s}',\boldsymbol{a}')\big]\big] \tag{7}$$

In tabular cases, dynamic programming algorithms, such as *policy iteration*, can be used to obtain the optimal solution for the problem [352]. *Policy iteration* is developed based on the *policy improvement theorem* and the *principle of optimality theorem* [352]. It is an algorithm that alternates between two phases *policy evaluation* and *policy improvement* until it converges to the optimal policy. The *value iteration* algorithm combines the two phases of policy iteration into one.

Q-learning [385] uses ϵ-greedy behaviour policy that generates actions to collect online data and deterministic greedy for final policy. The action-state value function is updated using the temporal difference (TD) error:

$$Q'(\boldsymbol{s}_t,\boldsymbol{a}_t) = Q(\boldsymbol{s}_t,\boldsymbol{a}_t) + \alpha(y_t - Q(\boldsymbol{s}_t,\boldsymbol{a}_t)) \ , \tag{8}$$

where the target value $y_t = r_{t+1} + \gamma(1-d)\max_{\boldsymbol{a}_{t+1}} Q(\boldsymbol{s}_{t+1},\boldsymbol{a}_{t+1})$ is estimated using the value of the target policy, and α is the learning rate.

Mnih et al. (2015) [255] propose *Deep Q-learning (DQN)* that extends *Q-learning* to solve high-dimensional/continuous state problems in practice, but where actions are still discrete. *DQN* leverages a deep neural network to approximate the state-action value function — the network takes true state vectors or observations as input and produces a vector of Q-values for each action. The Q-network is updated using the gradient of the temporal difference error as shown in Equation 8. Apart from the off-policy strategy used in Q-learning, it also utilises a replay buffer to learn from all the historical transitions to improve learning efficiency. Using such off-policy training can produce biased estimation of the Q-value on the current network. To mitigate the bias, it uses a target network, updated periodically from the current network with *polyak averaging* (soft update) [118] to estimate the target value. *Double Deep Q-learning (DDQN)* [371] further reduces the bias by choosing the next action using the current Q function and evaluating using the target one.

In order to apply *Q-learning* in continuous action space, *Normalised Advantage Functions* [121] choose a quadratic function to represent the advantage function for easier optimisation. Also, we can get the maximum Q-value using policy-network and optimise it with gradient ascent on expected Q-values. This starts to blur the boundary between value-based and actor-critic methods. Such algorithms include *DDPG* [231], *TD3* [106] and *SAC* [128]. These methods use mean square Bellman error (MSBE) to optimise the critic function:

$$\mathcal{L}(\mathcal{D}) = \mathop{\mathbb{E}}_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}',\boldsymbol{d})\sim\mathcal{D}}\left[\left(\hat{Q}(\boldsymbol{s},\boldsymbol{a}) - \hat{y}(\boldsymbol{s},\boldsymbol{s}',\boldsymbol{a},d)\right)^2\right] \ , \tag{9}$$

where the target value is $y(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{a}, d) = \mathcal{R}^{\boldsymbol{a}}_{\boldsymbol{s},\boldsymbol{s}'} + \gamma(1-d)\max_{\boldsymbol{a}'} \hat{Q}(\boldsymbol{s}', \boldsymbol{a}')$.

Deep Deterministic Policy Gradient (DDPG) [231] is an online off-policy MFRL value-based AC algorithm that learns a Q-function and a deterministic action network. It is trained off-policy by sampling experience from the replay buffer. *DDPG* optimises (1) MSBE for training the critic and (2) the expected Q-value of the policy for training the actor. In *DDPG*, the target value y is calculated from the target Q-network with the action generated by the current policy. As a Polyak-average copy of the training network(where the new target network is the interpolation between old target network and current training network), target networks stabilise the learning by avoiding the Q-value chasing its own tail. *DDPG* uses Gaussian action noise to explored the environment. The application of *DDPG* in cloth-shaping tasks will be discussed in Section 4.1.4.

DDPG fails in some scenarios due to overestimating Q-values. *Twin Delayed DDPG (TD3)* [106] mitigates this issue by introducing clipped *double Q-learning*, delayed policy updates and target policy smoothing. In delayed policy updates, the policy and the target networks are updated less often than the value network. In target policy smoothing, *TD3* augments the target action with noise to prevent the policy from exploiting the value function errors. *Soft Actor-Critic (SAC)* [128] is formulated in the framework of MaxEnt-RL, so it is covered in Section 3.5.1.

3.1.2 Policy-Gradient Methods

In CDO manipulation, motor-skill policy gradient based on *VPG* [358] and advanced policy-gradient *TRPO* [62, 60] are applied in dressing in simulation (Section 4.3.4). This section will introduce the policy-gradient methods from original *REINFORCE* [353] to *PPO* [326].

Policy-gradient is a direct method to achieve RL objective (see Equation 5) by optimising the parameterised policy π_θ using its gradient against the objective:

$$\nabla_\theta \mathcal{J}(\pi_\theta) \cong \mathop{\mathbb{E}}_{\tau\sim d^{\pi_\theta}}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)\Phi_t\right] \;, \tag{10}$$

where $\mathcal{J}(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim d^{\pi_\theta}}\left[\sum_{t=1}^{T} r_t\right]$, and Φ_t can be (1) return of a trajectory $R(\tau)$ [353], (2) reward-to-go $R_t(\tau) = \sum_{t'=t}^{T} r_{t'}$, (3) action-value function $Q^{\pi_\theta}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, (4) reward-to-go minus a baseline b_t (usually the value function $V^{\pi_\theta}(\boldsymbol{s}_t)$), or (5) the advantage function $A^{\pi_\theta}(\boldsymbol{s}_t, \boldsymbol{a}_t) = Q^{\pi_\theta}(\boldsymbol{s}_t, \boldsymbol{a}_t) - V^{\pi_\theta}(\boldsymbol{s}_t)$. In this section, we will talk about policy-gradient algorithms based on these variants.

The choice of Φ_t and the way of estimation of the values are the keys to reducing the variances and biases of the policy gradient. The common techniques includes causality trick, baseline subtraction and multi-step estimation. Besides, policy gradient is an on-policy approach due to the distribution of the expectation in the gradient, which introduce biases in the off-policy policy-gradient formulation. Another problem in the policy-gradient methods is that small parameter space updates can cause big policy space changes that make the training unstable. Some of these variants of the techniques lead to policy-gradient actor-critic algorithms.

REINFORCE [353] is the original policy gradient method for solving RL problems in low dimensional tabular cases, where the true equality holds in Equation 10, i.e., $\Phi_t = R(\tau)$. *REINFORCE* iteratively improves the policy using approximated policy gradient with sampled trajectories and the return of those trajectories:

$$\nabla_\theta \mathcal{J}(\pi_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\boldsymbol{a}_{t,i}|\boldsymbol{s}_{t,i})R(\tau_i) \;, \tag{11}$$

where N represents numbers of different sampled trajectories. In practice, this algorithm does not always work well because the optimisation suffers from high variance in the gradient.

Vanilla Policy Gradient (VPG) [353] updates the policy function based on the advantage function that is calculated by the difference between the reward-to-go $R_t(\tau)$ (causality trick) on trajectories generated by the latest policy π_θ and parameterised value function $V_\phi(\boldsymbol{s}_t)$ as a baseline b_t, so $\Phi_t = R_t(\tau) - V_\phi(\boldsymbol{s}_t)$ to reduce the variance in the gradient. The value function is updated using the mean-squared error between the value estimation and reward-to-go on all the states on the collected trajectories:

$$\phi^* = \arg\min_{\phi} \frac{1}{N \times T} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(V_\phi(\boldsymbol{s}_{t,i}) - R_t(\tau_i)\right)^2 \quad , \tag{12}$$

where ϕ^* represents the parameters of the optimal value function. As VPG trains a stochastic policy, action steps for exploration are sampled from the latest stochastic policy. However, there is a risk that the algorithm converges to local minima, as the policy becomes less random at the end of the training and encourages the agent to exploit states previously encountered. Also, a slight change in the parameter space can considerably modify the policy space. One can use first-order gradient descent to control the updating step on the parameter space, but we want to control the changes in the policy space. *Natural Policy Gradient* (NPG) [283, 282] resolves the issue by adding a constraint on the policy space:

$$\begin{aligned} \theta^* &= \arg\max_{\theta'} (\theta' - \theta)^\top \nabla_\theta \mathcal{J}(\pi_\theta) \\ \text{s.t.} \quad & D[\pi_{\theta'} || \pi_\theta] \leq \epsilon \quad , \end{aligned} \tag{13}$$

where D is the divergence measure between two distributions; usually, KL-divergence is chosen:

$$KL[\pi_{\theta'} || \pi_\theta] \approx (\theta' - \theta) F (\theta' - \theta) \quad , \tag{14}$$

where $F = \mathop{\mathbb{E}}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(\boldsymbol{a}_{t,i}|\boldsymbol{s}_{t,i}) \nabla_\theta \log \pi_\theta(\boldsymbol{a}_{t,i}|\boldsymbol{s}_{t,i})^\top\right]$ is the *Fisher-information* matrix [5]. After reformulating the optimisation problem using the Lagrange multiplier α, the update of the parameter becomes:

$$\theta' = \theta + \alpha F^{-1} \nabla_\theta(\mathcal{J}_\theta(\pi_\theta)) \quad . \tag{15}$$

Trust Region Policy Optimisation (TRPO) [325] tries to take the most significant possible improvement on policy parameters without causing performance collapse. To do so, it uses surrogate advantage that measures how policy π_θ performs compared to another policy $\pi_{\theta'}$. *TRPO* updates its policy using the surrogate advantage while keeping a constraint that indicates how different the new and old policies are allowed to be to avoid taking a big step:

$$\begin{aligned} \theta_{new} &= \arg\max_{\theta} \mathop{\mathbb{E}}_{\boldsymbol{a}\sim\pi_\theta, \boldsymbol{s}\sim\mathcal{P}} \left[\frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\theta_{old}}(\boldsymbol{a}|\boldsymbol{s})} A^{\pi_{\theta_{old}}}(\boldsymbol{a}, \boldsymbol{s})\right] \\ \text{s.t.} \quad & \mathop{\mathbb{E}}_{\boldsymbol{s}\sim\rho_{\pi_\theta}} \left[KL\left[\pi_\theta(\cdot|\boldsymbol{s}) || \pi_{\theta_{old}}(\cdot|\boldsymbol{s})\right]\right] \leq \epsilon \quad . \end{aligned} \tag{16}$$

Solving this objective is difficult in practice, so *TRPO* approximates the surrogate advantage and the mean of the KL-divergence using the Taylor series up to the first order and second order of the corresponding functions. The objective can be solved by optimising a quadratic gradient that resembles *NPG*. *TRPO* is applied in self-dressing and assistive dressing tasks in simulation (Section 4.3.4).

TRPO is conceptually and computationally challenging mainly due to the constraint in Equation 16. *Proximal Policy Optimisation (PPO)* [326] proposes a simpler objective:

$$\begin{aligned} \theta_{new} &= \arg\max_{\theta} \mathop{\mathbb{E}}_{\boldsymbol{a}\sim\pi_\theta, \boldsymbol{s}\sim\mathcal{P}} \left[\mathcal{L}(\boldsymbol{s}, \boldsymbol{a}, \theta_{\text{old}}, \theta)\right] \\ \mathcal{L}(\boldsymbol{s}, \boldsymbol{a}, \theta', \theta) &= \min\left(\frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\theta'}(\boldsymbol{a}|\boldsymbol{s})} A^{\pi_{\theta'}}(\boldsymbol{a}, \boldsymbol{s}), \quad \text{clip}\left(\frac{\pi_\theta(\boldsymbol{a}|\boldsymbol{s})}{\pi_{\theta'}(\boldsymbol{a}|\boldsymbol{s})}, \ 1-\epsilon, \ 1+\epsilon\right) A^{\pi_{\theta'}}(\boldsymbol{a}, \boldsymbol{s})\right) \quad . \end{aligned} \tag{17}$$

Several steps of gradient ascent can deliver the policy update, while clipping is a regularisation that makes the changes in policy less drastic, but it is still possible that the update will take a large step in the policy. There are many techniques to deal with this issue, but the update can be cancelled if the original constraint goes beyond a certain threshold. *PPO-Penalty*, on the other hand, subtracts the KL-divergence from the objective as a Lagrangian term. As an on-policy algorithm, *PPO* updates its policy on trajectories sampled by the latest policy. Furthermore, the value function update and sampling of the exploration steps are the same as those of *VPG* and *TRPO*.

3.2 Representation Learning

When developing an end-to-end robust skill controller, the quality of the latent representation directly affects the performance of the policy. The aim of representation learning in RL (RLRL) is to learn a good latent representation $\boldsymbol{z}$ from observations $\boldsymbol{x}$ or states $\boldsymbol{s}$ so that it can closely approximate the true state $\boldsymbol{s}$ of a POMDP/MDP. This helps to reduce the *curse of dimensionality* in MDP policy learning and, most importantly, plays the role of state estimation in POMDP settings. Learning good representation in a POMDP is crucial because it will help to improve the **learning and sampling efficiency** of the RL algorithm; it has been shown that learning policies from state-based input are significantly more sample-efficient than learning from observation [359, 208].

There are many candidates for good latent representation in RL. It has been argued that a good representation should be able to learn task-relevant information [412], preserve most information to predict the future [269] and capture the posterior distribution of the underlying explanatory factors for the observation [32].

Representation learning in DRL is builds heavily on work from the wider DL community. Common methods of representation learning comprise (1) data augmentation [405, 208, 209, 135, 340] (Section 3.2.1), (2) policy/value regularisation [405] (Section 3.2.1), (3) contrastive learning [208, 340] (Section 3.2.1), (4) input/latent mutual information [148], (5) maximising posterior distribution with dynamic [129, 129, 131, 417, 219, 331] (Section 3.3.2) or without dynamic learning [406] (Section 3.2.2), (6) predictive information [269, 7, 221], (7) Bisimulation [412, 50] and (8) asymmetric training by taking advantage of simulation [286]. So far, only data augmentation, contrastive learning and latent dynamic learning approaches have been applied in cloth-shaping domain (Section 4.1.4).

Most of RLRL falls under the umbrella of auxiliary tasks [169] that rely on an extra loss function on top of the RL objective. *Unsupervised Reinforcement and Auxiliary Learning (UNREAL)* [169] is one of the first such algorithms. *UNREAL* leverages two auxiliary tasks — pixel control and reward prediction — and jointly trains them with *A3C*'s original objective. This algorithm also uses an *Long Short Term Memory (LSTM)* layer [151] to perform latent state approximation from history observation and reward sequence.

The main challenge in POMDP settings is obtaining accurate state estimation from partial observation. We can use a "windowing" technique to concatenate small sequences of observations like in *DQN* [255]. We can also use a latent state $\boldsymbol{z}$ with recurrent neural layers to aggregate historical observations, such as in *Never Give Up (NGU)* [18] and *Agent 57* [17]. If using a dynamic model, we can further refine the state estimation by combining the prior estimation from the model and posterior estimation from the observations, such as in *PlaNet* [130], *Dreamer* [129, 131], *SLAC* [220] and *Muzero* [322], which works like a non-linear Kalman filter.

Shelhamer et al. (2016) suggest that DRL algorithms should spend considerable time on representation learning [333]. This encourages pre-training of the representation before training the exploratory RL agent. They can be trained with RL objectives simultaneously in an end-to-end fashion or train different parts of the agent alternatively throughout the online training. The representation learning techniques can be applied to offline data to train a good observation/state encoder $\mathcal{E}$ in an unsupervised and/or self-supervised way before interacting with the environment.

3.2.1 Data Augmentation

Data augmentation (DA), such as random flipping and cropping of images, are a type of up-sampling and regularisation methods that have demonstrated ability to improve data efficiency and generalisability of supervised computer vision tasks. In general, DRL's encoders suffer from observation overfitting [338, 277], so DA can also be used to improve **data efficiency** and **generalisation** ability of a DRL agent [209, 135]. It has been shown that *Random Shift* can overcome the overfitting problem of vision encoders [405]. We refer to Laskin et al. (2020) [209] for available DA functions and their comparison in the DRL domain. They experimentally show that *Random Crop* gives the best performance improvement.

DA improves the generalisation ability of the agent by letting it learn to extract observation features that matter to the task performance. It can be treated as a domain randomisation technique that adds variations to the observation space. However, optimisation of RL becomes increasingly challenging with the increasing variation on the observation due to the limited expressiveness of NNs [135]. It has been shown that DRL uses DA suffers from unstable training and poor performance [209].

DA is often applied with contrastive representation learning that further improves the data efficiency of downstream tasks by learning good latent representation [142, 55, 140]. As a sub-discipline of self-supervised learning, *contrastive learning* (CL) aims to create an embedding space where similar samples stay close while dissimilar ones are far apart. CL achieves its goal by automatically labelling positive and negative pairs and optimising a contrastive loss function. It has been successful in computer vision, natural language processing and reinforcement learning [215, 208].

Contrastive Unsupervised Representations for Reinforcement Learning (Curl) [208] utilises contrastive learning on the visual observations with data augmentation to produce a better latent representation of the state for the reinforcement learning algorithms. *Soft Data Augmentation (SODA)* [135] further improves **sample efficiency** and **stabilises** the RL optimisation by decoupling data augmentation from policy learning. It adds a soft constraint that maximises the mutual information between the augmented and non-augmented data, while underlying RL uses non-augmented data. Experimentally, they show that *Random Overlay* is the best data augmentation technique that improves the data-efficiency of pixel-based DRL. Note that *Random Overlay* [412] interpolates the observation with an unrelated image, and achieves *Bisimulation* in practice. By definition, two states are bisimilar if they share the same immediate reward and equivalent distributions over the next bisimilar states [207, 116].

Apart from using contrastive learning, we can also use augmented observation to regularise the RL objective function. *Data-regularised Q (DrQ)* [405] leverages both data augmentation and explicit regularisation term that is added to the value-based RL objective. Since DA introduces variations in observation space, value functions become unstable to train, although DRL ultimately benefits from it. The explicit regularisation term is used to stabilise the training. Generally, it follows three steps: (1) Data augmentation with small random shifts. (2) Average target Q-value over several image transformations. (3) Average Q-value estimation over several data transformation.

Curl and *DrQ* have been examined in cloth-shaping tasks in simulatioan [234], which will be discuss in Section 4.1.4.

3.2.2 Posterior Latent Distribution

Before introducing the RLRL techniques that maximise the posterior latent distribution, we briefly cover the variational autoencoder that is the common tool used to achieve such objectives.

Variational autoencoder (VAE) is an implementation of *Amortised Variational Inference* that approximates the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ with a stochastic inference function $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ [186] that learns the latent feature distribution of the data. *VAE* can also be treated as a generative model (GM) trained with variation inference; it has a generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ that can generate new data points from latent distribution:

$$\text{GM}(\mathcal{D}, \boldsymbol{x}) \doteq \arg\min_\theta D\big[p_\mathcal{D}(\boldsymbol{x})||p_\theta(\boldsymbol{x})\big] \ , \tag{18}$$

where $\mathcal{D}$ represents the dataset. KL-divergence is a common choice for learning the generative objective. Expanding the objective with KL-divergence, optimising the generative objective is equivalent to maximising the expected marginal log-likelihood of the observable variables over the data distribution:

$$\text{GM}_{KL}(\mathcal{D}, \boldsymbol{x}) \doteq \arg\max_\theta \mathop{\mathbb{E}}_{p_\mathcal{D}(x)}\big[\log p_\theta(\boldsymbol{x})\big] \ . \tag{19}$$

In the case of *VAE*, the marginal log-likelihood term can be optimised by maximising the *Evidence Lower Bound (ELBO)*.

$$\log p_\theta(\boldsymbol{x}) \geq \mathop{\mathbb{E}}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big] - KL\big[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big] \doteq \mathcal{L}_{VAE-ELBO}(\boldsymbol{x}, \phi, \theta) \ . \tag{20}$$

For simplicity, normal distribution is often chosen for the prior distribution $p(\boldsymbol{z})$, so that the inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ infers the means $\mu_\phi(\boldsymbol{x})$ and standard deviations $\sigma_\phi(\boldsymbol{x})$ for the latent variational distribution. Subsequently, the objective for *VAE* becomes:

$$\begin{aligned}\mathcal{L}_{VAE-ELBO}(\phi, \theta) = \mathop{\mathbb{E}}_{p_\mathcal{D}(\boldsymbol{x})}\Bigg[&- \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},I)}\Big[\frac{1}{2\sigma^2}||p_\theta\big(e_\phi(\boldsymbol{x},\boldsymbol{\epsilon})\big) - \boldsymbol{x}||_2^2\Big] \\ &\underbrace{- \frac{1}{2}\Big(||\mu_\phi(\boldsymbol{x})||_2^2 + ||\sigma_\phi(\boldsymbol{x})||_2^2 - d - 2 < \log\sigma_\phi(x), 1 > \Bigg]}_{KL\big[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big]}\end{aligned} \tag{21}$$

where the latent state $\boldsymbol{z}$ is sampled with *Monte Carlo estimation* with *reparamerisation trick* $e_\phi(\boldsymbol{x}, \boldsymbol{\epsilon}) = \mu_\phi(\boldsymbol{x}) + \sigma_\phi \odot \epsilon, \ \ \epsilon \sim N(\boldsymbol{0}, I)$.

VAE is an important tool used in many RLRL approaches for learning maximum posterior latent distribution. Yarats et al. (2019) [406] leverage β-VAE [147] on the single-step observations $\boldsymbol{x}$ to infer a latent state representation $\boldsymbol{z}$. The objective of this process is the same as the VAE's, but it works as an auxiliary task on top of a base RL algorithm:

$$\mathcal{L}_{\beta-VAE}(\phi, \theta) \doteq \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}}\Bigg[\mathop{\mathbb{E}}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\big] - \beta KL\big[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big]\Bigg] \ . \tag{22}$$

The underlying RL algorithm is trained using the inferred latent state, and the gradients are never shared between the β-VAE for learning the representation space and the RL objective. It has been shown that the stochastic nature of a β-VAE and the non-stationary actor's gradients affect the algorithm's performance and that **compact representation** is essential. Hence, Yarats et al. (2019) use a deterministic autoencoder with L2 regularisation on the latent space and the decoder parameters (known as RAE approach [115]) and allow the vision encoder to be updated only by reconstruction loss and value loss:

$$\mathcal{L}_{RAE} \doteq \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}}\Bigg[\log p_\theta(\boldsymbol{z}|\boldsymbol{x})\big] + \lambda_1||\boldsymbol{z}||^2 + \lambda_2||\theta||^2\Bigg] \tag{23}$$

Such posterior latent distribution can be better estimated with a latent dynamic model, such as *Recurrent State Space Model* [130] (Section 3.3.2) and *Control as Inference* in POMDP [219] (Section 3.5.1).

3.3 Model-based RL

Model-based reinforcement learning (MBRL) [257] can be subdivided into three main types: those that **plan using a learned model/known model** (Type I), **learn policy/value with imagined trajectory induced by learned model** (Type II), or **implicitly learn the transitional structure of the domain** (Type III). In general, MBRL is more data efficient compared to MFRL algorithms, and it generalises better when using large and diverse data [162]. Type I MBRL is heavily applied in cloth-shaping (Section 4.1.4) and assistive dressing tasks (Section 4.3.4), while we are aware of no application of Type II and III MBRL in CDO domain.

Planning, or trajectory optimisation, algorithms [35] usually generate local solutions for a subset of state space. They often need to get access to the dynamic model of the environment, i.e., either a known dynamic $\mathcal{P}$ or a learned dynamic $\hat{\mathcal{P}}$. The planning direction, including forward, backward and bi-directional; and planning budget, such as searching depth and breadth, are important factors of different planning algorithms. We can classify these algorithms into black-box planning and gradient-based planning. Shooting methods are only optimised on actions, while collocation methods optimise both actions and states. Mostly utilised planning algorithms in DRL domains are *Model Predictive Control (MPC)* [46], *Linear Quadratic Regulator (LQR)* [31] and *Monte-Carlo Tree Search (MCTS)* [41]. We refer readers to Moerland et al. (2020)'s review [256] for a more detailed survey about planning algorithms, i.e., Type I MBRL.

Type II MBRL algorithms are an order of magnitude more data-efficient than their model-free counterparts. However, they suffer from suboptimal asymptotic performance mainly due to the epistemic error [99]. This type of MBRL trains RL objectives by training on imagined trajectories induced by the policy and the dynamic model. In POMDP settings, such dynamic learning provides a good latent representation for value/policy learning. Type III MBRL algorithms, also known as implicit MBRL, do not train a dynamic model explicitly, but implicitly learn the transitional structure instead. Such algorithms often learn value equivalent models and/or learn to plan [257].

In the case of an unknown or expensive dynamic, the simplest way to learn an approximated dynamic model $\hat{\mathcal{P}}$ is to train it using **supervised learning** [174]. Transitional samples can be induced by a **base policy** π^{base}, such as a random policy. This approach generally does not work well in practice, especially in high-dimensional settings [154, 155]. This is mainly due to the **distributional shift** problem, i.e., $\rho_{\pi_{\hat{\mathcal{P}}}}(\boldsymbol{s}_t) \neq \rho_{\pi^{base}}(\boldsymbol{s}_t)$ [272]. This problem is exacerbated when an expressive model is adopted. However, it can be partially mitigated by iteratively gathering samples from the policy $\pi_{\hat{\mathcal{P}}}$ and training a new model $\hat{\mathcal{P}}'$ using the **aggregated dataset**. In training, it is better to collect transitions that lead to expected high rewards under the uncertainty of the dynamic [162]. We can also utilise **demonstration data** to reduce the distribution shift even further, as it makes the model learn the important part of the state space [162].

To avoid risky and erroneous actions caused by the **compounding error** in the model, we can choose the first action generated from the trajectory from the policy $\pi_{\hat{\mathcal{P}}}$. Then, we **replan** for other future actions. The most direct way to reduce the compounding error in planning is to (1) use **multi-step prediction** in model training [99, 404] and (2) have a **short planning horizon**.

In a high dimensional setting, it is possible to use *Variational Inference* with an NN-parameterised dynamic model to account for the aleatoric uncertainty in the system [130]. Apart from the distributional shift, epistemic uncertainty is partially caused by **overfitting** the model to the training data. MBRL algorithm that plans through learned models can exploit the epistemic uncertainty in training and testing time. This is known as **model exploitation** [162]. Other than the techniques we have discussed above, another way to mitigate model exploitation is to incorporate the model **uncertainty estimation** for policy generation. An alternative approach to tackle model exploitation is to fit **local models** and distil the knowledge of local policy to a global policy [417]. **Partial observability** is another factor that accounts for epistemic uncertainty. There are many attempts, such as *RSSM*-based algorithms [130, 130] and *Visual MPC* algorithms [81], to leverage recurrent state-action transitions for better representation learning and state estimation (Section 3.3.1).

3.3.1 Observational Dynamic Models

The ability to predict future outcomes of actions for a given event is a key component of human decision-making. The human brain incorporates a complex embedded representation of the physical and causal rules of the world [125]. With such a predictive ability, an RL agent can plan either based on observations or the latent space. Furthermore, such a model can provide a good latent representation for policy learning, which helps improve **data efficiency**. Agents can further benefit from the observational dynamic model by learning from imaginative trajectories. However, learning such a dynamic observational model is difficult for an RL agent due to the **aleatoric uncertainty** and **partial observability** of the environment.

The most popular approach in such settings is anticipating vision outcomes, since vision captures rich information about spatio-temporal interactions between objects. Ebert et al. (2018) [81] discuss the important roles of the visual dynamic model, i.e., action-condition video prediction model, in POMDP settings of robot control and propose a framework called *Visual MPC* that does planning on the predicted pixels of the future states.Note that *Visual MPC* using SV2P [16] and SVG [73] has been investigated in the cloth-shaping domain (Section 4.1.4). Apart from the two challenges mentioned above in dynamic learning in general, visual dynamic learning becomes more challenging when there are **occlusions** occurring among objects. Memory-based approaches, such as recurrent networks, must be adopted in such settings. Furthermore, good encoding representation from videos is also challenging due to the high dimensionality and variability of the visual space.

A great amount of research has gone into action-free and action-conditional video prediction. The goal of video prediction (VP) [270] is to predict future frames of a video $\boldsymbol{x}_{1:T}$ from a given history context frames $\boldsymbol{\xi} = \boldsymbol{x}_{t-m:t}$. It requires understanding causal events in the video [220]. The most direct objective of VP is to maximise the marginal likelihood of the future frames while conditioning on the context frames.

The time dimension of a video can be exploited because the consecutive frames are semantically coherent. This leads to many transformation-based architectures. Occlusions, lightning and camera perturbation mainly cause the variability between images. A wide range of techniques, including regularisations, complex loss functions, adversarial training, usage of flow maps, explicit transformations, separation of motion from the context and semantic and instance segmentation, have been proposed to improve prediction quality in deterministic VP. Refer to Oprea et al.'s (2020) [270] review for more details about deterministic VP. However, the aleatoric uncertainty of the system requires us to develop stochastic models, where *Variational Inference* [413] plays a critical role. In chronological order, popular and successful stochastic architectures include SV2P [16], SAVP [220], SVG [73], VRNN [58] and Hier-VRNN [49].

The common problem of variational architectures is that they produce blurry images due to trying to average possible future predictions while using simple MSE loss. Adversarial generation has been proposed to reduce this problem [220] by rejecting low-quality generated frames. Castrejon et al. (2019) [49] argue that naive variational encoding suffers from **underfitting** issue, especially on large datasets [391]. They suggest using a more flexible expression of variational distribution to improve the fitting ability. Opera et al. (2020) [270] claim that the blurry imagery is due to the loss function of the reconstruction that leads **regression-to-the-mean** problem [245]. Hence, they suggest investigating more powerful loss functions. Another way to resolve this problem is to use pixel-autoregressive models, which are considered slow [372, 178, 300].

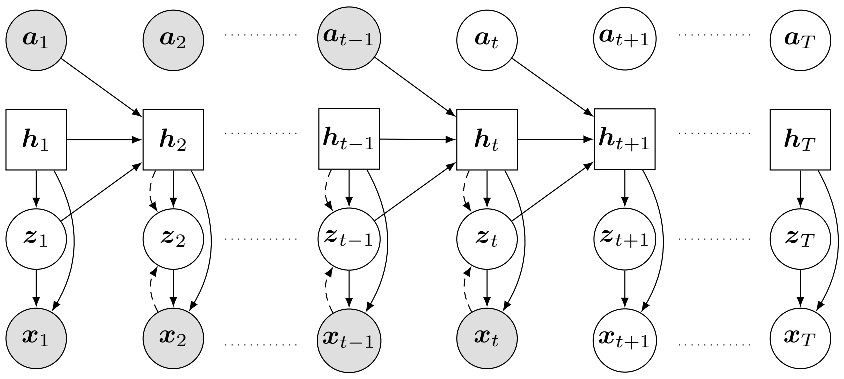

Figure 1: PGM of *RSSM* [219]

3.3.2 Latent Dynamic Models

There have been numerous latent dynamic models that use vision prediction for representation learning developed in the RL community. Hafner et al. (2019) [130] propose the *Recurrent State Space Model* (RSSM), which is a probabilistic model that predicts latent states of an environment with a latent dynamic model and refines its prediction based on a sequence of observations. Figure 1 shows the *Probabilistic Graphical Model* (PGM) for the *RSSM* model. As the actual state of the environment $\boldsymbol{s}$ is often inaccessible, *RSSM* is defined in the POMDP setting with the following latent state dynamic:

$$\begin{aligned} \text{Recurrent dynamic model} \quad & \boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1}), \\ \text{Representation model} \quad & \hat{\boldsymbol{z}}_t \sim q(\hat{\boldsymbol{z}}|\boldsymbol{h}_t, \boldsymbol{x}_t), \\ \text{Transition predictor} \quad & \tilde{\boldsymbol{z}}_t \sim p(\tilde{\boldsymbol{z}}|\boldsymbol{h}_t), \end{aligned} \tag{24}$$

where $\boldsymbol{h}$ represents the deterministic latent representation of state $\boldsymbol{s}$, $\tilde{\boldsymbol{z}}$ represents the prior of the stochastic latent state, and $\hat{\boldsymbol{z}}$ is the posterior induced by the deterministic latent state $\boldsymbol{h}$ and the current observation $\boldsymbol{x}$.

Built upon *RSSM*, *PlaNet* [130] aims to learn the latent dynamic that can generate accurate vision and rewards from a prior latent distribution for *MPC* planning. This is achieved by maximising the *ELBO* between prior and posterior latent states and the maximum likelihood of reconstruction of the observation and reward. Thus, it also includes the following network models:

$$\begin{aligned} \text{Observation predictor} \quad & \hat{\boldsymbol{x}}_t \sim p(\boldsymbol{x}|\boldsymbol{h}_t, \boldsymbol{z}_t) \quad , \\ \text{Reward predictor} \quad & \hat{r}_t \sim p(r|\boldsymbol{h}_t, \boldsymbol{z}_t) \quad . \end{aligned} \tag{25}$$

Thus, while conditioning on the action, the objective becomes:

$$\begin{aligned} \mathcal{L}_{PlaNet-ELBO} = \sum_{t=1}^{T} \Bigg(& - \underset{q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}, \boldsymbol{a}_{1:t})}{\mathbb{E}} \Big[\log p(\boldsymbol{x}_t|\boldsymbol{z}_t) \Big] \\ & + \underset{q(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1}, \boldsymbol{a}_{1:t-1})}{\mathbb{E}} \Big[KL\big[q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}, \boldsymbol{a}_{1:t}) || p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_{t-1} \big] \Big] \Bigg) \quad . \end{aligned} \tag{26}$$

PlaNet employs *Gated Recurrent Units (GRU)* [59] as the backbone of the latent dynamic model. *MPC with Cross Entropy Method* produces the policy at run time by unrolling and maximising the accumulative future rewards from the latent dynamic distribution. *PlaNet* uses

MSE as the loss function to learn observation reconstruction and reward prediction. It uses diagonal Gaussian distribution to model the variational variable. In order to reduce the prediction error, it also utilises multi-step loss, called *overshooting* in the paper, on the variational distribution and the reward prediction.

Dreamer [129] is a AC Type II MBRL method that uses latent dynamic model *RSSM* [130] for representation learning. The architecture and objective of the latent dynamic model are directly inherited from *PlaNet*'s [130]. Besides learning representation through maximising the marginal likelihood of the observation from the inferred latent state, they [131] also introduces another objective — maximising the mutual information between the latent state and observation [123] using contrastive learning. *DreamerV2* [131] leverages categorical latent state space representation and KL-balance to learn the latent dynamic model. Also, instead of minimising MSE, it tries to maximise the log-likelihood of the observation reconstruction and reward prediction.

Combining the latent dynamic learning and MaxEnt-RL (Section 3.5.1), Lee et al. (2020) propose *Stochastic Latent Actor-Critic (SLAC)* [219] that manages to train policy and dynamic network jointly in an end-to-end manner. The latent model objective of SLAC is:

$$\mathcal{L}_{SLAC-Model-ELBO} = \mathop{\mathbb{E}}_{\boldsymbol{z}_{1:T}\sim q}\left[\sum_{t=1}^{T} -\log p(\boldsymbol{x}_t|\boldsymbol{z}_t) + KL\big[q(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{z}_{t-1}, \boldsymbol{a}_t)||p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{a}_t)\big]\right] . \tag{27}$$

This corresponds to the model part of the *ELBO* of *Control as Inference* on POMDP. The difference between *SLAC*'s latent model and *RSSM* is that *SLAC* does not require deterministic latent variables and recurrent neural networks (RNNs).

Seo et al. (2022) [331] propose *Action-Free Pre-Training from Videos (APV)* that utilises the **pre-training and fine-tuning** paradigm in RL. They pre-train an action-free latent video prediction model from diverse domains and then leverage the pre-trained representation to learn a domain-specific latent dynamic model. They achieve this by stacking the action layer on top of the action-free model.

PlaNet has been applied in cloth-shaping literature (Section 4.1.4), but it has so far shown poor performance, likely due to the blurry observation prediction [233, 155]. We are not aware of any applications of *Dreamer*, *SLAC* and *APV* in any of the CDO domains.

3.4 Multi-task RL

Caruana et al. (1997) [48] states that "multi-task learning is an approach to inductive transfer that improves generalisation by using the domain information contained in the training signals of related tasks as an inductive bias. It does this by learning tasks in parallel while using a shared representation; what is learned for each task can help other tasks be learned better". Broadly speaking, multi-task reinforcement learning (MTRL) [377, 44] refers to a general RL agent that can perform tasks in different domains. In more narrow terms, MTRL is equivalent to **goal-conditioned reinforcement learning** (GCRL), where the agent can perform a given objective across tasks within a dynamic-invariant task family. GCRL, especially *HER* [9], is investigated heavily in cloth-shaping (Section 4.1.4).

The major challenge in multi-task learning is the **distraction dilemma**, where the necessary capability for performing individual tasks competes for limited resources within a single learning system. Often, such competition leads to **winner-take-all** problems where the performance of one task dominates the learning resources. Multi-task learning using gradient descent methods suffer from the **gradient interference** [409] problem while optimising shared parameters against multiple loss objectives. Scalability is closely related to the two major weaknesses of RL algorithms. First, training RL algorithms usually takes a considerable amount of time and needs more data samples to converge to an acceptable result. Second, an RL agent that is trained on a specific task can only be used on that same task [377].

Parallel multi-task learning and **continual learning** are the two main types of multi-tasks learning. Parallel multi-task learning trains on all tasks at once, while the tasks are trained sequentially in continual learning. Continual learning also suffers from gradient interference and the competition for resources, and it also suffers from **catastrophic forgetting**. Some methods, such as *Progressive Neural Network* [308], are proposed to mitigate this problem. Adding side connections to learned features in the previous tasks, *Progressive Neural Network* allows the agents to not only learn a series of tasks experienced in sequence without catastrophic forgetting, but simultaneously possess the ability to transfer knowledge from previous tasks to improve convergence speed in a new task. In both types of setting, **transfer learning** is the central idea to make the system perform across different domains by sharing common knowledge [418], with the aim to achieve faster learning and better performance in a new domain.

Transfer learning techniques usually work well if the source and target domain features are similar. In the RL, transfer learning techniques, such as fine-tuning [127, 8, 103, 204], domain randomisation [294, 309, 365, 170] and domain adaptation [370, 109, 369, 89] shows good performance while the underlying dynamic and states of MDPs are similar. Apart from transferring the dynamic, sharing a certain level of representation abstraction is especially crucial in POMDP MTRL [124]. Value functions can also contain useful knowledge to transfer because they implicitly model the structural relation between the state space and goal states. Similarly, successor features of the domain [24] can also be leveraged to achieve transfer learning in RL. One of the key aspects of MTRL is that the agent should develop a library of general skills that can be reused across various related tasks [377, 280, 307], which is also known as skill transfer in robotics (Section 1.3).

Parallel MTRL is one of the most common learning approaches [144]. This approach regards multiple MDPs as a single joint MDP. It leverages a single critic combined with different actors that resemble *A3C* algorithms [254], but the actors solve the different tasks in parallel. SOTA algorithms which leverage such techniques are *IMPALA* [87] and *PopArt* [144]. Instead of learning a central actor, *Policy Distillation* [307], *Actor-Mimic* [280], *DnC* [114] and *Distral* [361] leverages a central policy network.

In GCRL, the goal works like a contextual parameter that only alters the reward function for each MDPs (Section 1.2). Goal-condition reward functions are often negatively related to the distance between the current state and the goal for training the agent in a self-supervised manner [9, 53, 75, 176, 232, 344, 91]. Such formulation leads to universal value functions [318], also known as goal-condition value functions. In Robotics, GCRL is the tool to solve robust skills policy. More interestingly, it can be used to improve the sampling efficiency of an RL agent [275] by associating with the idea of deep exploration [242], which is covered in Section 3.5.

Andrychowicz et al. (2017) [9] propose *Hindsight Experience Replay (HER)* that trains goal-condition Q-function based on *DQN* and goal-condition policy function based on *DDPG*. This algorithm can be applied to off-policy RL algorithms on multi-goal/single-goal sparse-reward task settings. In *HER*, the replay buffer stores the transitional data with the original goal of an episode but also with some modified goals. It was shown experimentally that *HER* has vast improvement compared to its base algorithm in many hard environments. The challenging part of this algorithm is recalculating the sparse-reward function for the newly sampled goals. It means that the true state of the steps in the latest episode has to be stored temporarily unless the reward can be calculated based on the observations.

Eysenbach et al. (2021, 2022) [91, 92] propose *Contrastive RL* that solves GCRL using contrastive learning (Section 3.2.1). The inner product of the feature of the current state-action pair with that of the goal configuration can be regarded as the universal value function. They train the agent with *InfoMax* [148] on the universal value function by sampling positive state samples for a state-action anchor from the future trajectory and negative state samples from the other trajectories. Compared to *HER*, *Contrastive RL* is easy to implement and has fewer components.

3.5 Exploration

One of the most important components of a RL agent is its **exploration** strategy of the **behaviour policy**, i.e., the policy the agent used to collect transitional data during training. If the agent explores insufficiently, it may not find highly rewarded states, and its policy will be suboptimal. If it finds highly rewarded trajectories but keeps exploiting them, it may also converge to a suboptimal solution because it may miss other highly rewarded trajectories [4]. The balance between exploration and exploitation directly influences the sampling efficiency of an algorithm that, in turn, affects the learning efficiency, the upper bound of its performance, and the training stability. A better exploration strategy also leads to robust behaviour, as it has been through various states throughout the training. For detailed surveys on this topic, please refer to [248, 6, 206].

Without considering the future outcome in a long horizon task, a good exploration strategy is one that (1) explores states that might lead to better reward gains, known as optimistic in the face of uncertainty and/or (2) explore states that are novel and give more information of the environment, known as intrinsic motivation. These two may be dual problems, but with the latter, the agent can explore the environment independently without an extrinsic reward signal. Some fundamental myopic exploration strategies are derived from *Multi-armed Bandit* (MAB) formulation [212], then scaled to more complex MDP/POMDP settings. MAB-derived strategies focus more on the information about the action, and they can be classified into greedy soft policy, *Upper Confidence Bound* (UCB) [14] and *Thompson Sampling* (TS) [364] approaches.

In more complex settings (high-dimensional or continuous MDP/POMDPs), simple exploration approaches suffer from large state-action space, as well as sparse, delayed and deceptive rewards [403], where the latter case is known as the hard-exploration problem [82]. Apart from being unable to explore every possible state-action pair, such long-horizon MDPs have causal structures that some states will not appear unless the agent takes specific trajectories. In a continuous action setting, the simple strategy that only explores the surrounding of the policy output with adding noise will often leads to local optima. POMDPs introduce another layer of challenges related to accurate estimation of the environment state. Higher quality representations can lead to better exploration [173, 238].

Exploration is a huge field in reinforcement learning literature. DRL community focuses on optimistic methods, including intrinsic motivation [13, 320], MaxEnt-RL (Section 3.5.1) and sampling-based methods [275, 276, 15] based on TS for tackling exploration in complex MDP/POMDP settings. In addition, demonstration learning, i.e., fine-tuning the RL policy combined with behaviour cloning, is another way to reach asymptotic performance with a small amount of data. Other attempts include injecting parameter noise [288], adversarial self-play [343] and state marginal matching [222, 138].

Ecoffet et al. (2019) [82] point out that the two direct consequences of myopic approaches, such as intrinsic motivation, are (1) the agent always forgets the highly rewarded region in the state space (**detachment**) and (2) the agent cannot robustly return to such states if remembered (**derailment**). Solving these problems requires a good global exploration strategy [42] that can better balance long-term and short-term environment information [206]. In the literature, such global exploration is also known as **deep exploration** [275]. The deep exploration method considers both myopic information gain and the long-term causal outcome of an action for future learning. Ecoffet et al. (2019) [82] propose a SOTA algorithm called *Go-explorer* that incorporates a memory buffer to store highly rewarded states and uses imitation learning to strengthen its ability to go to these regions to do further exploration.

Another approach to achieve deep exploration is to use goal-based strategies [265, 290], where the important states are proposed to guide the exploration of the agent. These approaches need a heuristic goal generator and a lower-level exploration strategy for exploring to achieve the goals and conduct further exploration once a goal is achieved [235]. This approach is related to skill discovery and robust skill policy learning in robotics, and Eysenbach et al. (2018) [90] present a self-supervised goal-based exploration algorithm by discovering diverse skill sets without any external reward signals [90].

In the continuous-time dynamic control domain, the exploration problems become even more challenging because the agent needs to decide how to discretise the time-space for applying a policy for a given state [249, 267]. Short time intervals allow learning a better policy, but it compromises the sample efficiency. Longer time intervals improve the sample efficiency, but the policy is more likely to become trapped in a local optimum. A good exploration strategy in continuous-time settings needs to dynamically decide the time intervals for balancing the trade-off between the sample efficiency and the learning performance [267].

Assessment of exploration strategy in a complex MDP/POMDP can also be tricky. It is usually assessed in terms of sampling efficiency, i.e. how many environmental steps an agent needs to reach asymptotic performance. Additionally, the robustness of the agent is a second metric to assess the exploration performance, by testing whether the agent can react optimally to a wide variety of states. *Montezuma's Revenge* in Atari games [29], *Deepmind Lab* and *Vizdomm* [182] are common environments to test these qualities.

3.5.1 Maximum Entropy RL

A type of optimistic exploration is the entropy-regularised RL agent derived from the MaxEnt-RL framework, where the entropy of a stochastic policy provides the intrinsic reward. Human behaviour is not optimal most of the time. The trajectory from the starting and end states can vary around the optimal trajectory. Instead of learning deterministic policy, maximum entropy reinforcement learning (MaxEnt-RL) [422] incorporates *Maximum Entropy Principle* to learn a distribution of policies whose mean is close to the optimal policy. This idea is critical to the exploration strategy of RL algorithms, as well as to the development of inverse reinforcement learning.

The concept of using maximum entropy in control theory was first introduced by Ziebart et al. (2008) [422] and reformulated by Levine (2018) [225]. Built upon MDP, both frameworks model the controlling problem as an inference problem. This provides the opportunity for modelling sub-optimal policies with stochasticity that often occurs in human and animal behaviour that encourage exploration and robust manipulation in a single RL task and facilitate faster training in transfer learning settings. It also provides an interesting interpretation of the reward function and possibly sheds light on the pathway to reward design. The objective is defined as

$$\text{MaxEnt-RL}(\mathcal{M}, \alpha) \doteq \pi^*_{soft,\alpha} = \arg\max_{\pi \in \Pi} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho_\pi} \Big[r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha \mathcal{H}(\pi(\cdot \mid \boldsymbol{s}_t)) \Big], \tag{28}$$

where α is the temperature scalar to interpolate between the original RL objective and MaxEnt-RL Objective.This objective can be understood as optimising RL objective the under the augmented reward function:

$$r^+_{soft,\alpha}(\boldsymbol{s}, \boldsymbol{a}; \pi) = r(\boldsymbol{s}, \boldsymbol{a}) - \alpha \times KL[\pi(\boldsymbol{s}) || \mathcal{U}] \ \ . \tag{29}$$

Hence, it is a type of intrinsic motivation exploration strategy where the intrinsic bonus is derived from the uncertainty of the policy.

By introducing binary optimality variables in high dimensional continuous MDP settings, Levine shows that MaxEnt-RL can be derived with control as probabilistic inference with known deterministic dynamics in tabular cases and variation inference without the knowledge of dynamics. This led to the MFRL method soft Q-learning [127] and the actor-critic method *Soft Actor-Critic (SAC)* [128]. Following from MaxEnt-RL, or *Control as Inference* framework, on POMDP, *Stochastic Latent Actor-Critic (SLAC)* [219] improve upon *SAC*'s objective by incorporating latent dynamic learning for learning good representation to learn soft value and policy functions efficiently. This algorithms achieve asymptotic performance using end-to-end learning. *SAC* has been investigated in cloth-shaping domain (as described in Section 4.1.4), but we are not aware of application of *SLAC* in the CDO domain.

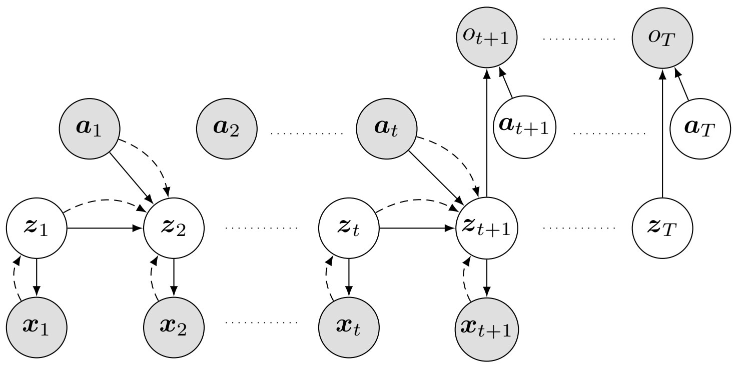

Figure 2: PGM of *Control as Inference* in POMDP [219]

3.5.2 Demonstration Learning

The most practical way to bypass sample efficiency problems in complex MPD/POMDP is to use demonstration trajectories. These approaches usually use BC methods (Section 2.1) to initialise the policy with demonstration data and leverage online collected trajectories to fine-tune its policy with the reinforcement learning objective. Demonstration learning (DemoL) is often used in robotics applications. SOTA DemoL algorithms are *DQfD* [145], *DDPGfD* [374] and *Q-filtered BC* [264]. In trajectory-level DemoL, a trajectory generated by trajectory BC controllers (Section 2.1.2), such as *DMP*, is usually refined by trajectory optimisation algorithms like *LQR* and *Policy Improvement with Path Integrals (PI2)* [54]. In CDO manipulation, DemoL algorithms are mainly used in cloth-shaping (Section 4.1.1) to reduce the exploration complexity and assistive dressing for safe trajectory control (Section 4.3.4) tasks.

Building upon *Prioritised Duelling Double DQN* [143], *Deep Q-Learning from Demonstrations (DQfD)* [145] is pre-trained only on the demonstration data using temporal difference and supervised losses. The losses include a one-step double Q-learning loss, a multi-step double Q-learning loss (similar to *A3C*), a supervised large margin classification loss and L2 regularisation on the network's parameters. Note that these losses are also applied in the second phase of the training on the demonstration data. Inspired by [287], the supervised loss function is defined as:

$$\mathcal{L}_{DQfD-SL}(\hat{Q}, \boldsymbol{a}_{demo}) = \max_{\boldsymbol{a}\in\mathcal{A}} \hat{Q}(\boldsymbol{s}, \boldsymbol{a}) + l(\boldsymbol{a}_{\text{demo}}, \boldsymbol{a}) - \hat{Q}(\boldsymbol{s}, \boldsymbol{a}_{\text{demo}}) \ , \tag{30}$$

where $l(\boldsymbol{a}_{\text{demo}}, \boldsymbol{a})$ is a margin function; it is 0, when $\boldsymbol{a}_{\text{demo}} = \boldsymbol{a}$, otherwise it is a positive constant ϵ; it makes the Q-value of the other actions at least a margin inferior to that of the demonstrated one. Then, the agent starts online learning with its pre-trained policy, which updates its value function with a mixture of demonstration and collected data. Demonstration data is preserved throughout the training, and a bonus is added to the weights of the demonstration data to increase the possibility that the replay buffer will sample them.

DQfN has outperformed its pure RL and pure IL counterparts on most of the Atari games. As with *DQN* (Section 3.1.1), this algorithm can only be applied to the discrete-action domain. *DDPG from Demonstrations (DDPGfD)* [374] is proposed for continuous cases and can also deal with sparse rewards. In *DDPGfD*, demonstration data is preserved throughout the training in a prioritised replay buffer. The priority of a transition comprises its latest TD error, actor loss and bonus factor for demonstration data. The sparse reward setting uses a mixture of one-step and multi-step loss to train the critic. L2 regularisation is applied to both actor and critic networks. Prioritised replay is used, which is essential for sparse reward settings. Multi-step return loss assists the propagation of the sparse reward along the trajectory.

Built upon *DDPG* [231] and *HER* [9], Nair et al. (2018) [264] propose a method called *Q-filtered Behaviour Cloning (QfBC)* that also applies behaviour cloning loss on the demonstration data to solve long-horizon, sparse-reward and goal-condition tasks. Nevertheless, this method keeps two different replay buffers instead: one is for the demonstration data and another for the trial data. Notably, it does not utilise priority replay buffers. At each update step, the algorithm samples a certain amount of data from both replay buffers. The update losses are similar to *DDPGfD* except that it also uses Q-filtered behaviour cloning loss on the demonstration samples for updating the actor, which is defined as follows:

$$\mathcal{L}_{QfBC} = \sum_i ||\pi_\theta(\boldsymbol{s}_i) - \boldsymbol{a}_i||^2 \left[Q(\boldsymbol{s}_i, \boldsymbol{a}_i) > Q(\boldsymbol{s}_i, \pi_\theta(\boldsymbol{s}_i))\right] \ , \tag{31}$$

where $[\cdot]$ is a Boolean expression function, which evaluates to 1 when its operand is true, and 0 otherwise. They also reset some trial episodes using the intermediate states and goals from demonstration episodes. The initial states of the resets are sampled from any intermediate states in a demonstration episode. As *HER* is also utilised, the goal state for these reset episodes is chosen to be the goal state of the same demonstration episode.

4 CDO-Manipulation Systems

Manipulating CDOs with robotic systems can be traced back to 47 years ago when Murray (1975) [262] design the first ply-picking end-effector for improving the productivity of garment factories.

Humans perform different CDO tasks in different ways. For a cloth-shaping task, we tend to focus on effective points, such as the contour and corners, on the article to grasp. If an effective point is hidden, we have good intuition to unravel the cloth to find these points. At the end of the task, we can smooth the wrinkles on the cloth by spreading our hands and stretching the corners in the opposite direction. For rope manipulation, such as insertion, knot tying and untying, we have foreknowledge to know where to grasp along the rope regarding the strains of the rope. We depend on our vision system for insertion tasks, and we can only depend on haptic sensors and motor skills to finish rope tying and untying tasks. As for bag manipulation, we can detect the hem of the bag for grasping effective points to open it. While lifting it, we can estimate the amount of force we need to hold the bag in our hands without tearing it. In dressing tasks, we usually ignore the deformation of the garment and mainly focus on whether our limbs go into the correct openings of the garments. We mainly rely on our haptic system to sense if we perform the task correctly and safely.

The most fundamental challenge in manipulating CDOs is that they have many degrees of freedom, which leads to complex dynamics and severe self-occlusion. The complex dynamic of a CDO introduces a major obstacle for manipulation because different parts of the cloth move differently with respect to their individual internal forces. There are aleatoric uncertainties about the cloth's deformation, meaning that the cloth doesn't always deform the same way under the same action in the real world. The complex dynamic of CDOs makes it hard for analytical model building, dynamic learning and exploration for RL algorithms. On the other hand, self-occlusion can hide effective key points for manipulation. This makes it hard to extract the state of the cloth and presents challenges for engineering reward functions and automating the goal evaluation for complex CDO manipulation tasks. Another major challenge is grasping. A two-fingered gripper is sufficient for most of the tasks in this domain. Nevertheless, roller end-effectors are also utilised for effective and safe grasping, and multi-fingered grippers are adopted for more dexterous manipulation such as tying a bowline knot [357].

In this section, we will talk about four prevailing task families in the CDO domain: cloth-shaping, rope manipulation, bag manipulation and dressing. We will discuss the individual challenges of these tasks and how the perception and control systems in the literature manage to tackle these challenges.

4.1 Cloth-shaping

Cloth-shaping is a crucial skill set for doing laundry in daily life [133]. We broadly define cloth-shaping as manipulating a single CDO, such as a rope, a towel, or a T-shirts, to a goal configuration. Narrowly, we define cloth as a square fabric made of any cloth-like material. The very canonical tasks of cloth-shaping is cloth flattening, where one or more end-effectors apply actions on a piece of square fabric to unfold it completely on a table. Humans tend to identify key points, such as corners and edges, on the article to perform cloth-shaping tasks. Sometimes, we take advantage of gravity, air dynamic and the inertia of the cloth to flatten it.

A more complex and important task in cloth-shaping is cloth-folding, which includes single-step folding (SSCF) and multi-step folding tasks (MSCF). SSCF includes diagonal folding, rectangular folding, side folding, one-corner inward folding, and other types of folding that can be achieved with one pick-and-place (P&P) action by a human. MSCF, on the other hand, includes cross folding, diagonal cross-folding, double-corner inward folding, all-corner inward folding, double side-folding and all other folding tasks requiring more than one P&P actions [386, 155]. Concerning the initial state, folding tasks can also be classified into folding from the flattened shape and folding from the crumpled form (FCF), where the latter is a more complex problem [234].

Flattening, folding, or other types of complex shaping can also be applied to other CDOs [251, 386], such as towels (rectangular fabric instead of square), T-shirts, trousers, shorts, dresses, coats and bags. The more complex the shape of the CDO, the harder the task will be. Many tasks in the cloth-shaping domain comprise several subtasks. For example, the system must flatten the object before actual folding to perform FCF. Similarly, Weng et al. (2022) [386] assign intermediate goals to perform MSCF tasks successfully. Furthermore, the nature of the task depends on the number of end-effectors. Usually, more end-effectors lead to more efficient manipulation. However, we need to consider extra constraints to avoid collisions among the arms. Because of this, comparison across a different number of end-effectors can be unfair.

4.1.1 Challenges in Cloth-shaping

Many real-world failures in cloth-shaping are caused by grasping deficiencies of the system. These deficiencies include (1) inaccurate grasp-point estimation, (2) misgrasping the target point on the cloth, (3) grasping multiple layers of the fabric if there is a fold and (4) rigid damage on the end-effector caused by hard surface while grasping the fabric. It is hard to develop such a system in simulation because almost none of the robotic and deformable object simulator can accurately simulate the interaction between the gripper and the CDO [327, 234, 155].

Secondly, it is hard to define the reward function and goal evaluation automatically for MSCF tasks because the final state of the cloth in these tasks is characterised by a high degree of self-occlusion. It also requires reasoning about complex spatial relationship between the current and goal observation in pixel space.

Thirdly, there are many applications of DRL in this domain, but they often suffer sampling efficiency due to the complex and enormous state-action space of cloth-shaping. Robust velocity control is still an unsolved challenge in this domain [244, 234, 171].

4.1.2 Perception in Cloth-shaping

Finding the most effective grasping point on a hanging article is important for successfully stretching the cloth in the air. Hamajima and Kakikura (2000) [134] propose a handcrafted hem-line detection system from captured shadows and outlines of an article. Kaneko and Kakikura (2001) [180] propose *Aspect Models* to keep track of the grasping points in an article. Kita et al. (2009) [188] match the mesh representation of a hanging T-shirt with a pre-simulated candidate template. Maitin-Shepard et al. (2010) [241] leverage depth discontinuities induced from the optical flow between frames for border classification. They apply the *RANSAC* algorithm [102] for image-to-3D localisation of the grasping points. Bersch et al. (2011) [34] create an estimated mesh representation of the target article using fiducial markers and depth sensor, from which it generates grasping candidates based on the partitions segmented by fold-lines. The fold-lines can be extracted using the shadows on the article. Cusumano-Towner et al. (2011) [68] model the grasping states of the article with HMM that provides prior to aligning the contour of the simulated article with the actual one. Doumanoglou et al. (2014) [77] leverage *Hough Forests* [108] for point estimation to grasp the key points in the article, and they refine the estimation using interactive perception by formalising the process with POMDP. All these methods try to find a pair of key points on the contour of the article that can be grasped to stretch without any misgrasping and twisting.

Finding effective key points based on existing wrinkles and folds is crucial for flattening an article on the table. Willimon et al. (2011) [389] use the *Harris Corner Detector* [137] on the cloth mask (a binary feature map for segmenting cloth in a scene) and filter the candidate grasping corners that connect directly towards the peak of the cloth without encountering a fold. To detect wrinkles, Yamazaki and Inaba (2009) [398] propose to apply *Gabor filtering*. Ramisa et al. (2012) [296] leverage χ^2*-kernel Support Vector Machine* to generate a heatmap for graspability based on wrinkles on the cloth. Sun et al. (2013) [345] detect the wrinkles on the cloth by clustering points using *K-means* and *Hierarchical Clusterings* algorithm using inout from a depth sensor. Sun et al. (2015) [346] further develop the description of a wrinkle and improve the precision of the detection by fitting piece-wise *B-Spline* surface to the depth map.

The contour of the cloth is often extracted for cloth folding. Miller et al. (2011, 2012) [250, 251] propose a set of parameterised polygon models as a geometrical representation that includes skeleton and folding lines for different types of garments. Their perception system tries to fit these template models on a flattened garment on the table. Stria et al. (2014) [342] and Doumanoglou et al. (2016) [79] match the contour of an article to its corresponding polygonal model [341] that includes important landmarks, such as corners, shoulders, armpits and crotches.

For tasks involving multiple garment types, the system needs to classify among different articles. Kaneko and Kakikura (2001) [180] propose a handcrafted classifier with a set of hand-designed features based on the geometrical shape of the garments. Osawa et al. (2006) [273] use a set of the hanging-length of garments as features to classify different garments. The features are extracted from binary cloth masks using edge-detection techniques. Willimon et al. (2011b) [388] conduct a similar classification with hand-engineered features using the nearest neighbour algorithm by comparing against a database of known items. Doumanoglou et al. (2014) [77] use *Random Decision Forest* [39] to recognise the garments from depth sensors.

Almost all the RMSs that operate the cloth on the table apply top-view cameras, with most using depth sensors. Such setting captures most of the feature information of a fabric while it is resting on the table. Another benefit of using a depth camera is that it is colour invariant [345, 386]. Hoque et al. (2022a) [155] also shows that RGB-Depth (RGB-D) produces the best prediction quality on the dynamic visual model compared to using just RGB or depth.

Optical flow is usually adopted to estimate motion in videos. In cloth-shaping, it provides a good representation of the relationship between the current and the goal image. Weng et al. (2022) [386] leverage optical flow to achieve flattening and various types of folding on both fabrics and T-shirts. Mesh representation of cloth can bring topological information into planning for grasping and flattening.

Mesh plays a significant role for keeping track of the state of the article, especially when there are occlusions in the visual input. It also provides an opportunity to automatically check the performance of multi-step folding tasks. Arnold et al. (2021) [10] leverage a neural network to infer probabilistic mesh representation from depth sensors. Lin et al. (2022) [233] and Huang et al. (2022) [161] propose a learning-based mesh reconstruction methods based on GNN from top-down observation.

4.1.3 Classical Control in Cloth-shaping

Hamajima (1998) [133] propose a manipulation flowchart for laundry folding on various garments. It includes steps from picking pieces up from a pile, garment classification, flattening and folding to putting them into the corresponding drawer. Maintain-Shepard et al. (2010) [241] provide the first autonomous system to execute the pipeline on a pile of towels. Together, Doumanoglou et al. (2014a, 2014b, 2016) [77, 78, 79] and Stria et al. (2014a, 2014b) [341, 342] also accomplish the whole pipeline [79] on a pile of different garments.

Before 2018, there were two main streams of research in cloth flattening: gravity-based and pick-and-place/drag (P&P) approaches. For removing the final wrinkles on the article, the system can still use P&P [345], but spreading is a more effective action primitive [79], where one arm fix the cloth from sliding and the other sweeps to an effective direction.

Gravity-based cloth-flattening action primitives stretch the article in the air and place it on the table [134]. This requires dual arms to complete the task. Hamajima and Kakikura (2000) [134] suggest grasping the hemline induced by the shadow on the hanged article using a roller end-effector and grasping the lowest point of the article when there are no such shadows. Osawa et al. (2007) [274] propose a handcrafted system that iteratively grasps the lowest point of a cloth until the convergence of the hanging length, then the bilateral arms stretch the cloth in the air. Salleh et al. (2007) [312] propose an analytical method to grasp the second point by sliding on the edge of cloth from the first grasped point using an inchworm gripper. Kita et al. (2009) [189] propose an analytical method for grasping an optimal point on a mesh representation [188] of a T-shirt from a hanging position. Maitin-Shepard et al. (2010) [241] improve the pipeline of laundry folding task [180] by including error recovery from misgrasping and twisting of the towel. Cusumano-Towner et al. (2011) [68] generate the grasping strategy using a shortest-path planning algorithm where intermediate mesh states are induced by simulation. Ramisa et al. (2012) [296] propose to grasp the article based on a graspability heatmap for initial grasping by hanging the article in the air, while Doumanoglou et al. (2014) [77] choose to grasp the estimated key points in the article.

Pick-and-place/drag action primitive achieves cloth-shaping on the table, which can be conducted only by one gripper. The difference between pick-and-place and pick-and-drag is that the second set of parameters for pick-and-place is place position, while for the latter it is the displacement vector. Willimon et al. (2011) [389] propose a heuristic approach to flatten a cloth on the table after applying a prescribed motion phase. If all the detected corners are connected to the peak region of the cloth, then a corner will be dragged away from the cloth's centre. Otherwise, the system assumes the cloth has a fold, and a corner will be dragged toward the centre for the convenience of later steps. Sun et al. (2013) [345] propose a heuristic approach to remove wrinkles on the fabric. The system grasps a cloth corner and drags it orthogonally to the orientation of the biggest wrinkle on the cloth. Sun et al. (2015) [346] continue to leverage the same strategy on fabric and T-shirts but use dual arms.

In terms of cloth-folding tasks, Osawa et al. (2006) [273] accomplish various types of garment folding with a prescribed motion by controlling a flip board using bilateral arms. The action primitives, flipping and rotating the board, are delivered by analytical models. Bell (2010) [27] demonstrate folding a t-shirt using the *Japanese method* without regrasping. Berg et al. (2010) [33] propose a gravity-based folding method *g-fold* that works on the geometrical representation of garments and achieves successful bilateral folding on various garments with prescribed motions. Miller et al. (2012) [251] also complete garment folding using *g-fold* motion planning on a parameterised polygon model [250]. Bersch et al. (2011) [34] propose an open-loop planning dual-arm system that can fold a T-shirt from a crumpled position after it stretches the article in the air. Doumanoglou et al. (2016) [79] utilise *g-fold* in motion planning on the polygonal model of the garments. Li et al. (2015) optimise a *Bezier curve* [94] folding trajectory using the *Levenberg-Marquardt* algorithm with deformable simulation against a quadratic cost function.

P&P cannot erase wrinkles efficiently when the system has nearly reached the goal. In contrast, dynamic manipulation leads to a much quicker feedback cycle and self-correction on failures. Moreover, it can exploit the physical property of CDO to reduce the total operation time [171, 234, 126, 146], especially for the cloth-flattening task. Apart from velocity control, one can also use inertia and air-dynamics to control the cloth. Ha and Song (2022) [126] propose pick-and-fling action primitives and a learning system *FlingBot* to achieve cloth flattening using dual arms. On the other hand, Xu et al. (2022) [395] propose a learning-based system *DextAirity* based on pick-and-blow action primitives that blow air under the article while a dual-arm grasping its corners.

4.1.4 Data-driven Control in Cloth-shaping

Applications of DRL and DIL methods on cloth-shaping have been explored since 2018. The adopted methods are BC (Section 2.1) [328, 327, 386, 362], MFRL (Section 3.1) [244, 392, 171, 126, 146, 234, 146, 224] and MBRL (Section 3.3) [402, 155, 237, 10, 234, 233]. So far, MBRL and BC have achieved better performance than MFRL methods [327, 155, 386, 233].

Most of the methods mentioned above adopt classical P&P actions defined on the pixel space. Few attempt more low-level velocity control to conduct dynamic manipulation [244, 234, 171, 146, 126]. One advantage of using such abstract action is that it shortens the necessary prediction window in planning and leads to much lower search complexity in the MFRL algorithms. It helps interpret the agent's policy at each action step. However, as the agent only perceives the environment before and after the long action step, this strategy has a long control feedback cycle. This makes manipulation more time-consuming and less robust to unexpected scenarios. In other words, agents cannot react immediately to interruptions, miss-grasping the fabric and miss-controlling from the underlying planning algorithms.

Discretising P&P action space is a common technique in CDO literature. For example, Lee et al. (2021) [224] discretise the drag strategy by putting different orientations and pulling distances into bins. Even though such action space simplifies the problem and leads to better training, it is less flexible during run time and harder to generalise to other tasks. Another technique that leads to better P&P policy learning is the separation of the inference of pick and place positions. Wu et al. (2019) [392] suggest using *Maximum Value of Placing* that selects the best picking position after choosing the best placing position. Weng et al. (2022) propose *Fabric Flow Net* that infer the pick-condition place policy [386].

Cloth-flattening and cloth-folding can be trained jointly using a uniform goal-condition data-driven approaches [224, 155, 171] thanks to their identical underlying dynamics as well as the similar state, action, goal and policy representations. Some approaches [224, 146, 171, 327] trained MFRL agents with *HER* [9], a self-supervised technique to train agents in sparse reward environment and generalise across the task family, to improve the data efficiency. Some approaches [155, 402] leverage the difference between current and goal observation to generate reward signals for MBRL methods. Although Yan et al. (2020) [402] only focus on the cloth-flattening task, the objective of minimising the distance between the current and the goal latent state can apply to all cloth-shaping tasks. Arnold et al. (2021) [10] and Hoque et al. (2022a) [155] attempt to apply this goal-condition policy representation at the mesh level with the attempt to solve more complex cloth-shaping tasks. In addition, some of the mentioned BC methods train a goal-condition *Transporter Network* [327, 411, 237] or flow network [386, 76] individually to improve data efficiency yet with P&P action primitives.

The complex and enormous state-action dynamic of the cloth makes it hard to train a policy representation in BC, a dynamic model in MBRL and a value function in MFRL algorithms. Thus, demonstration data [244, 328, 171, 155, 327], corner-biased data [155, 386] (which biases the picking action towards corners and edges of the fabric) and other engineered data collection strategies [10, 233] are utilised to speed-up the training. Whilst only Lee et al. (2021) [224] collect real-world data, most learning-based methods operate in simulated environments, where some apply domain randomisation [365] to achieve *Sim2Real* [386, 126, 155, 402]. The simulators used for cloth-shaping tasks include *Nvidia Flex* [234, 126, 386], *Pybullet* [327], *Mujoco* [392, 402], *SOFA* [95, 171] and *Unity with Obi* [387]. Note that *Blender* is only used for rendering [328, 155].

To tackle the exploration problem, some data-driven methods set key points of the fabric as the observation space [244, 171, 237]. In contrast, others attempt to reconstruct the corresponding mesh to guide the manipulation [10, 155, 233]. Policy noise [130], demonstration data [244, 311], specially engineered data [156, 386], *HER* [9], MaxEnt-RL objective [392, 146, 234] and advantage-weighted loss exploration term [311] are leverage to overcome the exploration problem for DRL applications.

The application of MBRL in cloth-shaping tasks suffers significantly from model exploitation and compounding error. It often requires training the model with a large amount of data (more than 100k observation-action pairs) [155, 402, 237]. Ma et al. (2021) [237] propose *G-DOOM* that is a graph-based dynamic model based on key points to reduce compounding error, while Yan et al. (2021)'s *Contrastive forward modelling (CFM)* [402] trains latent dynamics for planning. Hoque et al. (2022a) [155] propose *Visual Foresight Modelling* that leverages variational video prediction models *SV2P* and *SVG* within the framework of *Visual MPC* [81] for mitigating model exploitation. Hafner et al. (2019)'s latent dynamic model *Deep Planing Network (PlaNet)* [130] has been examined in detail in the literature [402, 237, 234], but it does not show good results as it does in other rigid-object continuous control domains. The reconstructed observation from the visual model is fuzzy [402, 155], which makes planning based on reconstructed vision hard because the edges and corners of the clothes are not clear. Lin et al. (2022) [233] also claims that learning a latent representation loses the detailed information of the target cloth, such as folds and wrinkles. Moreover, particle-wise learning-based dynamic models, such as *MeshGraphNets* [284] and *GNS* [314], achieved incredible results using GNN. *Visible Connectivity Dynamics (VCD)* [233] applies such mesh models on the visible part of the cloth to achieve more precise planning.

Data augmentation functions, such as scaling and rotation, on the observation are used to improve data efficiency for end-to-end cloth-shaping control systems [224]. Action noise can also be applied to the policy output for improve the data-effeciency, but only with small perturbations, as the next state of the cloth is susceptible to the applied action [224]. Domain randomisation, such as randomising the background colour of the table [154, 402], is used for transferring the simulation-trained end-to-end policies to the real world (Section 1.4).

In robotic applications using RL, the reward and the goal signal of cloth-flattening are usually assigned based on the coverage area of the fabric in its vision input [155]. Rewards of cloth-folding are usually given by the difference between the current and the goal vision inputs [155]. For simple cloth-shaping tasks, the difference between the current and goal observation at the pixel space helps both MFRL and MBRL algorithms train successfully [155]. *CFM* [402] and *G-DOOM* [237] leverage the distance between the current and goal latent representation. Also, particle-wise distance estimation from the observations between the current state and the goal state shows improved performance on two-step folding tasks [155]. Hoque et al. (2022a) [155] propose an effective dense reward function based on the coverage difference between two consecutive states for the cloth-flattening task. The reward function also rewards performance when reaching 92% coverage and penalises the misgrasping failures and out-of-boundary scenarios. The reward and goal signals of some MSCF tasks, such as triangle-folding and square-folding, cannot be directly extracted from vision input. It makes those tasks more challenging compared to other MSCF tasks, although a system can divide them into smaller subtasks to adapt to vision signals [386].

There are several benchmark environments in the cloth-shaping domain. Lin et al.'s (2021) [234] *SoftGym*, based on *Nvidia Flex* [239], includes rope stretching, cloth flattening, cloth folding and cloth placing tasks. The same authors also provide the learning performance of *Oracle MPC*, *SAC-DrQ*, *PlaNet* and *SAC-Curl* using velocity control signals as action space. Seita et al. (2021) [327] propose *DeformableRaven* based on *PyBullet*, and provide execution performance of a goal-conditioned *Transporter Network*. Hoque et al. (2022b) [156] propose a real-world cloth-flattening and cloth-folding benchmark based on *GoogleReach* [390], which is a cloud physical workcell equipped with a robot arm and a table that can be accessed remotely through internet.

Currently, there is no standard automatic way to evaluate the performance of a cloth-shaping system. Human perception is mainly used to assess if a physical trial is successful. Mean and standard deviation of particle-wise distance towards its goal have been used in many cases [386]. The difference between current and goal observation also has been utilised to evaluate a method [402, 156]. This metric is beneficial for automating the evaluation in physical trials, but it cannot capture all information in more complex problems than one-step folding tasks. The return value of an episode [234] and the reward of the last episode steps can also measure the performance of a method. As the reward functions of those methods differ, it is not easy to compare various methods. For cloth-flattening tasks, normalised improvement [233] and normalised coverage [233, 155] are the most reliable metrics that can be automatised both in simulation and reality. However, this metric does not explicitly include information on wrinkles on the cloth, which is crucial for evaluation when a trial is nearly successful. As secondary metrics, the number of steps and inference time are used to measure a system's effectiveness.

4.2 Rope Manipulation

In this review, we define the rope-manipulation task family as tasks executed on cloth-like linear deformable objects (CLDOs), such as ropes, wires and threads. Manipulation tasks on CLDOs include grasping, shaping, knot-tying, knot-untying and wrapping objects. The task family also encompasses insertion and suturing, which have applications such as in surgical robotics and assembly-lines in factories [313]. Rope manipulation also has important applications in climbing, dressing and decoration. Rope manipulation is a set of harder problems than cloth-shaping. Rope-shaping tasks, where a robot puts a target rope into a certain shape without any crossing, have similar properties as cloth-shaping. We will not go into the details about rope-shaping, as they are often covered in data-driven cloth-shaping literature which we have discussed in the last section. This subsection will mainly discuss knot-tying and knot-untying applications in the literature.

Knot tying (KT) encompasses (1) tying a particular knot on a single rope (knots), (2) tying to connect two or more ropes (bends) and (3) tying the rope to an object (hitches). The simplest and most common knot one can tie is an overhand knot. Knots for performing single-rope tying (SRT) also include double overhand, figure-of-eight, masthead, reef and sheepshank knots, as well as bowline, bowknot and Ume-knot [67]. The ones used for multiple ropes connecting (MRC) include square and sheet-end knots, as well as strop, harness and carrick bends. Moreover, the standard knots for typing a rope on an object (TRO) are half and clove hitches. Finally, some knots are used for decoration, such as sounding lines, cloverleaf knot and Ruyi knot.

KT depends on whether the rope is tied on the table or in the air. The number of end-effectors, their capabilities and the usage of extra tools can drastically change the nature of the task. The inverse process, knot untying (KU) is also a difficult problem for robotics that correspondingly covers (1) untying a particular knot on a single rope, (2) untying a knot that connects multiple ropes and (3) untying the rope from an attached object.

Humans mainly use three fingers (index, middle and thumb) to tie knots. The skills we use comprises of bending, twisting, holding and binding the rope [379]. We can even make knots without vision input after we hold them in our hands, and we check if the knot is tightened using interactive perception.

4.2.1 Challenges in Rope Manipulation

In addition to CDO's complex dynamic and self-occlusion issues, manipulation of CLDOs also suffers from another problem called self-symmetry. It means that a rope looks exactly the same from the start to end points and vice versa. RMSs in the literature usually use a topological representation that specifies the target rope's start and endpoints. It is hard for a perception system to consistently keep track of the two points due to the self-symmetrical property of a CLDO. Note that self-symmetry is not a problem for cloth-shaping, because the system does not have to keep track of the topological structure of the cloth to solve flattening and folding tasks.

KT becomes much harder if the target ropes are tangled at the beginning. In such cases, the agent must unravel the rope before tying the target knot. The direct challenge to untying a knot is to recognise its knot structure [236]. The robotic system may introduce more crossings to untie a knot. The crossing structure of a tangled rope can be more complicated than a knotted rope [380], which makes the search space of motion planning algorithms much larger.

Furthermore, MRC introduces more crossings while trying to tie/untie multiple ropes [376] than dealing with a single rope. It also presents more endpoints of ropes in the scene, which amplifies the self-symmetry problem and makes it difficult to keep track of the status of every specific rope.

4.2.2 Perception in Rope Manipulation

Most approaches in KT/KU literature choose to operate at the topological level, so the perception system needs to extract a topological representation (TR) from its sensory data. There are many types of TR used in the robotic literature. Morita et al. (2003) [259] leverage P-data [97], a collection of segments and intersections, as the intermediate representation to generate K-data [97], an ordered vector of intersections and segments from start to end points. Wakamatsu et al. (2004, 2006a) [379, 378] propose a sequential representation that combines the K-data and crossing information of the 3D-to-2D projected rope with (u, l) representing crossing over or under. This representation also includes $(+, -)$ [80] to denote two types of crossings related to the direction of the two segments at the crossing. This representation has become common in later KT/KU systems. Additionally, Matsuno et al. (2006) [247] further reduce detection error caused by the deformation of real-world rope using invariant knot properties. The common strategy for producing TR is to use the extracted intersection and segments of the rope from thinned binary images [153, 379, 378, 259, 397, 202].

For approaches that leverage imitation learning, the perception systems produce rope descriptor that annotates the corresponding locations on the rope among different inputs. Sundaresan et al. (2020) [350] use a supervised learning approach with NNs to learn such descriptors. Some systems use the behaviour cloning approach *Thin Plate Splines Trajectory Transfer (TPS-TT)* [323, 324, 217]. The perception system of these methods takes the current and goal image and produces a warp function that maps the corresponding points on the rope. Lee et al. (2014) create such a function with registration from point cloud data [217]. Huang et al. (2015) [160] improve the perception by adding RGB imagery as input and key-point segmentation as an intermediate representation. On the other hand, Suzuki et al. (2021) [355] train an autoencoder to learn the latent representation from RGB input and tactile sensors.

4.2.3 Classical Control in Rope Manipulation

Both KT and KU start from structural representation in a computer simulation using the knowledge of knot theory. The early KU approaches in topological simulation include random perturbation [336, 317], annealing schedule [230], energy minimisation with gradient descent [159] and motion planning [205, 380]. These are all energy-minimising methods. Ladd et al. (2004) [205] adopt RRT-inspired motion planning [213] to untie various topological knots by minimising an energy function. These energy functions encompasses geometric energy, Möbius energy, spring energy, electrostatic energy and minimum distance energy [336, 205]. Nonetheless, these functions are not suitable for KT. Wakamatsu et al. (2004, 2006a, 2006b) [379, 380, 378] introduce a general motion planning framework for both topological KT and KU using four Reidemeister moves [301, 379] without any energy function.

Inaba and Inoue (1985, 1987) [166, 167] leverage a stereo vision system for planning to tie a knot. Hopecraft et al. (1991) [153] suggest a graph representation that captures the topological structure of the rope. They also develop a high-level action grammar that can be decomposed into P&P action primitives with orientation changes of a two-fingered gripper to tie several different knots on the table. Wakamatus et al. (2004, 2006a, 2006b) [379, 378, 380] propose a sequential representation of the rope and replace the a motion planning algorithm to search for a sequence of primitive actions to achieve a goal configuration. The primitive actions based on knot theory are three *Reidemeister Moves* (RM) [301]. They are designed to manipulate a mathematical knot that is a loop with no ends. RM I achieves loop production/removing by adding/removing one crossing; RM II simultaneously adds/removes two crossings; and RM III moves a segment to the other side of a crossing.

Most KT systems act either on the table [153, 379, 378, 217, 160, 348] or in the air, [247, 397, 396, 373, 202, 355]. Similar to cloth-shaping, P&P action primitives are leveraged for the tasks performed on the table [153]. In KT, the agent usually requires more coordination between the two end-effectors to achieve the task. Exceptionally, Yamakawa et al. (2010) [396] achieve the overhand-knot in the air using a one-arm high-speed dynamic control system with the assistance of the gravity and rope's inertia.

For tackling physical ropes, Wakamatsu et al. (2004) [379] posit the fourth type of RM, while Matsuno et al. (2006) [247] propose continuous transformations, including expansion and contractions. Yamakawa et al. (2008) [397] suggest rope pulling and moving operations, which are similar action primitives to the RM IV. Furthermore, they also introduce a rope permutation primitive that can cancel rope deformation. On the table, a target knot can be tied and untied by these four types of RMs and rope permutation that can be delivered with P&P action primitives, but this process cannot guarantee the tightness of the knot. The grasping strategies of the rope directly influence the efficiency and success of the tasks. Wakamatsu et al. (2004) [379] recommend a set of grasping strategies for its primitive action w.r.t. the local crossing patterns.

Inaba and Inoue (1985) [166], Hopcraft et al. (1991) [153], [259] and Wakamatus et al. (2004, 2006a, 2006b) [379, 378, 380] have paved the fundamental methodology for classical KT/KU literature. The conventional way to develop a KT/KU system comprises of (1) defining action primitive based on knot theory and end-effector capability, (2) employing a visual perception system to extract a topological representation of the rope and (3) adopting motion planning with knowledge of knot-theory to finish a task. The appropriate choice of perception, where to grasp the rope and how to deliver the target primitive have remained long-standing problems for both lines of control methods. Subsequently, Saha and Isto (2006) [310] leverage a topological motion planar to achieve tying bowline, neck-tie, bow and stun-sail knots in simulation using bilateral robot arm with leading needles. Lui and Saxena (2013) [236] leverage a motion planning method to achieve a 77% real-world dual-arm untying success rate on a collection of different types of ropes and knots.

Pulling both ends of a rope is enough for tightening simple knots such as an overhand knot. This can be delivered by either fixing one end of the rope and pulling the other or grasping both ends and pulling them towards opposite directions. However, many complex knots used for connecting and decorations must be tightened at specific locations along the rope [382] by leveraging friction locks, i.e., locations where friction resits the rope against external forces. Bell et al. (2014) [28] and Wang and Balkcom (2016) [382] employ needle fixtures to achieve friction locks on knots for decoration. Checking if a knot is tight in robotics is still an open research area, even though answers have been proposed in analytical literature [23, 298, 11, 47].

4.2.4 Data-driven Control in Rope Manipulation

Apart from motion planning, LfO is another line of research in KT/KU literature. Morita et al. (2003) [259] introduce *Knot Planning form Observation* (KPO) that integrates control methods from LfO on the topological level of the rope, where the changes of the representation in the consecutive demonstrated observation produce a primitive action. Sundaresan et al. (2020) [350] achieve 66% real-world success on tying the overhand-knot by improving the descriptor perception system within the LfO framework.

Apart from LfO, Vinh et al. (2012) [373] use trajectory replay to achieve single-arm overhand-knot tying in the air. Kudoh et al. (2015) [202] also adopt a similar method to type a square knot on a cylinder using a three-fingered dual arms. Lee et al. (2014) [217] and Huang et al. (2015) [160] employ trajectory behaviour cloning method *TPS-RPM* [57] to transfer the demonstration policy after registering current observation to a keyframe, followed by trajectory optimisation to refine the suggested trajectory. They achieve overhand knot using dual arms on the table. Takizawa et al. (2019) [357] apply the same method to achieve overhand knot and figure-eight knot using a pair of three-fingered arms. Suzuki et al. (2021) [355] leverage the multi-modal deep behaviour cloning method to achieve a 95% success rate on in-air dual-arm bow knot and overhand knot in real-world trials.

Granne et al. (2020) [120] create a *Hierarchical Untangling from Learned (HULK)* system that suggests grasping points for untying action primitives. Sundaresan et al. (2021b) [349] propose a robust rope grasping system called *Local Oriented Knot Inspection (LOKI)* that is trained with behaviour cloning method using the synthetic data generated from Blender [65]. Sundaresan et al. (2021b) [349] present *Sensing Progress in Dense Entanglements (SPiDER-Man)* to recover from the error caused by *HULK*. Viswanath et al. (2021) [376] managed to untie multiple cables with their *Iterative Reduction Of Non-planar Multiple cAble kNots (IRON-MAN)* system that generates primitive actions to remove crossing on the cables. Combining *LoKi*, *HULK* and *SPiDERMan*, the system achieves an 80% success rate in real-world trials on untangling three cables.

There are many applications of IL in rope manipulation literature, but it is still not clear how to frame KT/KU in MDP to develop an RL controller. One of the challenges is reward shaping. Using P&P action primitives. Fan et al. (2022) [93] use discrete-action DRL algorithm *Deep Q-learning (DQN)* [255] that takes the embedded states as input and discretised grasping points and moving directions as action. They achieve a 54% success rate using single-arm for untying knots on the table. Their reward credits the crossings' removal while penalising the crossing's increment and ineffective operations.

4.3 Dressing

Worldwide demographic trends indicate that the portion of the elderly will increase considerably in the future. This implies we will experience a nursing shortage for elderly and disabled people. One of the robotics community's long-term goals is to develop such care-taking autonomous robots that improve the living quality and independence of people in need. A significant challenge of such a system is assisting humans with reduced mobility to dress.

Dressing tasks are the second most investigated topic in CDO manipulation literature. Apart from assistive dressing, self-dressing is another challenging domain that belongs to the dressing task family. A virtual self-dressing agent can provide autonomous dressing animation for film production. Although there are no significant real-life applications of self-dressing agents, they share many properties with assistive dressing systems in terms of representation and motor skills.

Assistive dressing stands for helping an immobile or partially immobile person to put on or take off various garments while ensuring the person is mentally and physically comfortable. *I-Dress* [1] is an ongoing project that aims to achieve assistive dressing. The garments include hospital gowns, jackets (with sleeves), vests (without sleeves), T-shirts, scarves, hats, trousers and other clothes. Assistive dressing is a long-horizon multi-step task that requires safe and reliable human-robot interaction. For instance, while helping a person put on a jumper, the agent needs to tuck the person's head into the jumper's hem, put the two individual arms into

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 December 2022 doi:10.20944/preprints202212.0305.v1

the corresponding sleeves, and then pull down and adjust the jumper. These subtasks, though they need to be executed sequentially, are usually investigated individually for various types of garments in the literature. Moreover, the assisted person can cooperate with the agent to finish the dressing. Many older adults have limited limb movement, and some patients may even shake unpredictably during the task. When a human helps another person to dress, they can use life experience to estimate the force exerted on the assisted person [84]. Similarly, we expect the system to provide smooth, predictable and small force-exerting action trajectories and to react accordingly to human posture and motion, as well as to avoid damaging the cloth.

Self-dressing refers to putting on and taking off various garments on a humanoid robot without damaging the robot's body and the garments. When humans put on a T-shirt they mainly rely on their body's haptic system. First, they put their head into the bottom of the T-shirt, then find and put the corresponding sleeves over their arms, put the head through the hem, and finally pull down and adjust the garment. While stretching our arms through the sleeves, we don't think about the complex interaction between the article and our limbs [61, 62]. Meanwhile, we are careful not to get snagged in or tear the cloth.

4.3.1 Challenges in Dressing

The ability to distinguish the inner surface of an article from the outer helps to avoid getting tangled in the article [61]. In self-dressing, it is specifically challenging for an agent with a tactile sensor to formalise and integrate such perception ability into the control procedure [62]. In general, self-dressing is easier than assistive dressing, but the two share some common challenges. First, topological and functional properties of garments are highly correlated [368]. Reliably finding the topological correspondence between the different parts of the body and the garment is a challenge for perception [191, 246]. The problem mainly lies in the occlusion of the garment by the torso and thedeformation of the article itself [192], where such deformation can be quite different when compared to cloth-shaping tasks [192].

Second, finding the effective grasping point and grasping strategy for dressing the corresponding parts is still an underexplored problem. Almost all of the literature experiment with the presumption of grasping a correct part of the article, while only Clegg et al. (2018) [62] integrate grasping as part of the skill learning using DRL. Third, dressing tasks generally deviate from common manipulation tasks. They heavily depend on tactile and haptic sensors to infer the progress of the task [62]: (1) force estimation of the cloth on the body is crucial for safe and comfortable dressing; (2) estimation of cloth-stretching degree helps to avoid damaging the cloth. It is not obvious how humans leverage such perception to perform a similar task, and this remains a difficult open problem in this domain [61].

In addition to the challenges mentioned above, the agent should also react accordingly to cooperation and the unpredictable motion of the user in assistive dressing. Unexpected user movements may lead to dressing failures or even pose risks to the user. The complex deformation of the cloth and its occlusion on the body makes human-posture tracking difficult [415, 60]. For example, occlusions can occur when the robot's arms, the garment and the human body are in close contact [112]. Such occlusion makes it complicated for the vision system to accurately observe the task state and predict the results of planned interactions. Furthermore, it is hard to test assistive dressing in the real world during the development stage, as mannequins cannot be easily actuated [175].

4.3.2 Perception in Dressing

Keeping track of the body-cloth relationship is essential for successful and safe assistive dressing. Tamei et al. (2011) [358] were the first to leverage topological coordinates [149] for modelling the body-cloth relationship, where the skeleton represents the torso and circles represent the opening part of the article. Chance et al. (2016) [51] and Yamazaki et al. (2014, 2016) [400, 401] propose handcrafted perception systems to extract the skeleton of the body. For robust real-time estimation of such topological relationships in the real-world trial, Koganti et al. (2013, 2014, 2015, 2017) have proposed a sequence of improvements in terms of perception [193, 194]. They adopt a data-driven latent model, *Gaussian Process Latent Variable Model* (GP-LVM) [214], to provide a prior for state estimation only from depth camera during testing [191, 192]. On the other hand, Twardon and Ritter (2016) [368] produce such priors using an analytical model on the boundary component of the cloth.

Force sensors can also help mitigate occlusion problems in visual perception. Gao et al. (2015) [111] combine vision and force sensors to estimate human pose and leverage GMM to model movement to overcome the occlusion problem. Kapusta et al. (2016) [181] employ haptic perception to infer the cloth-body relationship by fitting an HMM. Erickson et al. (2018) [85] track the body pose in real-time using capacitive proximity sensors. Zhang et al. (2017) [414] achieve latent posture tracking trained with GP-LVM on RGB-D and human arm posture data. Zhang et al. (2019) [415] improve the real-time posture tracking system using a probabilistic filtering method where the GP-LVM has only been trained on forces and the position of the end-effector. The initialisation of the posture is given by a depth sensor.

Such skeleton extractions and posture tracking can provide trajectories that avoid collisions between the end-effector and the body. Still, we also need reliable force estimation for safe and comfortable reactive manipulation. Erickson et al. (2017) [84] use recurrent neural layer LSTM [151] to estimate a force map injected on the human body from the force, torque and velocity of the robot's end-effector. Clegg et al. (2018) [62] use a haptic sensor as an observation space for DRL application to learn a reactive policy.

In addition to avoiding a large amount of force, the system also needs to detect success and different error states. Yu et al. (2017) [410] tuned the simulator to collect haptic data to train an HMM for predicting dressing outcomes. Chance et al. (2016, 2017) [51, 52] combined force sensors with inertial measurement unit (IMU) sensors to detect dressing errors using machine learning methods. Yamazaki et al. (2013, 2014, 2016) [399, 400, 401] propose a complex framework that uses optical flow from two consecutive images for detecting dressing outcomes. Yamazaki et al. (2014, 2016) [400, 401] utilise both visual and force inputs to improve the quality of such dressing state detection.

4.3.3 Classical Control in Dressing

In the realm of HRI, a conservative policy uses collision avoidance and compliant control [43] for reducing the force applied on the human body [228]. In assistive dressing, the common approach is to combine these two classes of algorithms by leveraging a motion planner to generate a collision-free predictive action sequence, then adopt compliant control with force sensors to move relative to body posture [319]. Yamazaki et al. (2014) [400] adopt an analytical trajectory planner considering collision avoidance for helping to put on trousers using dual arms. Klee et al. (2015) [190] leverage a sample-based motion planner to help a person put on a hat. Chance et al. (2016) [51] propose analytical trajectory planning for helping to put a target's arm into a jacket's sleeve based on a mannequin's skeleton. They use velocity control [211] and mitigated potential collisions [203] for safe human-robot manipulation. Gao et al. (2015) [112] add an iterative path optimisation algorithm using gradient descent to minimise the analytical force resistance estimation on a prescribed path.

Subsequently, Erikson et al. (2018) [86] employ capacitive proximity sensors to impose distance control between the end-effector and the human body on a prescribed trajectory. Zhang et al. (2017, 2019) [414, 415] present a latent posture tracking method for generating dressing trajectory. They utilised a hierarchical analytical controller to perform assistive dressing while reducing the force applied to the human. Li et al. (2021) [228] apply *Model Predictive Control* on a human dynamic model for safe motion planning [195] under the context of *Human-Aware Motion Planning* [210], i.e., collision avoidance or safe impact in the event of a collision. The systems proposed by Klee et al. (2015) [190] and Chance et al. (2016) [51] ask the target to repose if a trial fails and re-planned for a new trajectory. However, this can only work for people with a specific motion capacity. Yamazaki et al. (2014, 2016) introduce the ability to recover from failures automatically without repositioning the target human [400, 401].

4.3.4 Data-driven Control in Dressing

In the data-driven control domain, assistive dressing literature tends to use trajectory BC (Section 2.1.2) methods to generate initial trajectories then utilise Type I MBRL (Section 3.3) methods for safe and smooth execution of trajectory. Tamei et al. (2011) [358] adopt a RL framework [335] to tuck a mannequin head into a T-shirt's hem and its arm into T-shirt sleeves using dual arms. It initialises the "via-points" with demonstration trajectory and refines the trajectory using the policy gradient method [252]. Matsubara et al. (2013) [246] use the same RL framework for learning self-dressing T-shirts on a dual-arm robot, specifically putting both arms into the sleeves of the T-shirt. Based on a similar framework, Colome et al. (2015) [64] use imitation learning method *Dynamic Movement Primitive (DMP)* [165, 164] to initialise the robot trajectory to wrap a scarf around a human's neck using a single arm. Then, they use *PI2* [363] to refine its policy. Pignat et al. (2017) [285] formulate the task where a robot assists a human to put their arm into a jacket with *Hidden Semi Markov Model* (HSMM) [408] for encoding the demonstration trajectory using *EM* algorithm. Then, they leverage a *LQR* [31] to drive the robot to follow the generated trajectory from the forward messages of the HSMM [83]. Joshi et al. (2019) [175] broke the task down into three consecutive subtasks and appied *DMP* for dressing the arm and a *Bayesian Gaussian Process Latent Variable Model* (BGPLVM) for dressing the body. In this domain, the real-world demonstration trajectory is usually collected by directly controlling the robot arms with hand, known as kinesthetic teaching [358, 246, 285, 285].

Clegg et al. (2017) [61] use the DRL policy gradient method *Trust Region Policy Optimisation (TRPO)* [325] (Section 3.1.2) with curriculum learning [384] in simulation to learn a modular haptic feedback controller of self-dressing. Clegg et al. (2018) [62] then managed to learn a complete self-dressing task in simulation using DRL with a specially engineered reward function. The observation space includes the human's joint angles, garment feature locations, haptics, surface information and a task vector. Clegg et al. (2020) [60] use DRL and curriculum learning for training a simulated dual-arm robot and a human with various motor capabilities for wearing a hospital gown and T-shirt collaboratively. The observation includes sensorimotor information of both robot and human, force signal from the end-effector and target pose from the previous time step. The action space in both approaches is based on the positional signals of the joints [62, 60]. In a real-world trial, this system requires the human to wear sensory equipment on their body. They achieve *Sim2Real* transfer by calibrating the simulator with real-world data and scaling down the policy output.

Reward engineering is a hard problem in the dressing domain. Tamei et al. (2011) [358] and Matsubara et al. (2013) [246] design their reward function around the topological coordinate (comprised with writhe, centre and density) distance between the configuration of the region of interests of the torso and the garment. Colome et al. (2015) [64] suggest a reward based on penalising high acceleration and high estimated force. They also included another term indicating how well the scarf is placed in images. Clegg et al. (2017) [61] penalise the distance to the goal and the failure of the task. Clegg et al. (2018) [62] posit a self-dressing reward function that comprises progress reward, deformation penalty (for avoiding tearing the garment), geodesic reward and per-joint "rest pos" reward (where the user body is in the default setting) [62]. However, this reward is not safe for the human body in assistant dressing tasks, so Clegg et al. (2020) [60] added another term to reduce the force received by the human.

Conventionally, evaluation of assistive dressing is conducted on a mannequin during development time and on real humans for final evaluation [51]. Self-dressing can be developed and tested on the robot itself in the real world. [246]. However, such physical testing cannot provide a large volume of data for a confident conclusion of robust manipulation. Hence, simulation becomes an ideal place for developing and testing dressing approaches [410, 60, 84]. The most common simulator adopted in this domain is *Nvidia PhysX* [2]. Furthermore, the mannequin cannot provide unexpected or collaborative actions in development and final testing. Clegg et al. (2020) [60] suggest using simulation to create scenarios where humans have different levels of disability: dyskinesia, limited range of motion and muscle weakness. In the real-world trial, they employ another humanoid robot to replace a mannequin for creating unexpected and collaborative motions.

4.4 Bag Manipulation

Bag manipulation is a relatively new and least investigated domain among the four task families in the literature, so we will cover conventional and data-driven control methods together in Section 4.4.3. We characterise bags as 3D cloth-deformable objects that can contain items. More specifically, bags refers to 3D CDOs that have handles, while sack represents bags without handles. RMSs with bag-manipulation skills can help a human with grocery shopping and transporting heavy items.

The canonical task of bag manipulation is item loading (ILD) [362] that involves opening the bag (OB), inserting items (II) [327] and lifting the bag (LB). Lifting a bag with an item in it is also known as an item containing (IC) [330]. The general grasping ability of various items and the effective grasping strategy of the bags are crucial for the success of ILD. II can be more complex if there are other bags for distraction [327]. The next common task is bag moving (BM) [362], which includes lifting the bag, creating displacement and placing the bag at the target position. The third standard task is item unloading (IU), which requires opening the bag and picking the items out. There is another prevalent task called bag unloading (BU) [117] where the agent is asked to unload a collection of bags from a basket.

4.4.1 Challenges in Bag Manipulation

Compared to ropes and cloth, the deformation of bags is more complex, and the self-occlusion is much more severe for a RMS [330]. The complex dynamic between the rigid/deformable items with the bag [387] gives another layer of challenge for the accomplishment of ILD. During II, the perception needs to reason whether the objects are within the region of open contour [330, 362] of the bag. While conducting IC, the system also needs to reason if items will remain inside the bag under their interaction with the bag and gravity [330]. Effective grasping strategy on a sack, which takes advantage of the gravity and other factors, is still an open problem in this domain [330].

Lifting a bag with items inside it demands an accurate weight estimation so that the bag will not slip from the gripper, as is common when grasping a thin layer [330]. Sacks introduce further difficulties compared to bags as they don't present handles. They usually require special end-effectors, such as rollers [117], so the agent can effectively grasp and hold the sack without damaging the sack. Moreover, bags can have different shapes and sizes, and the centre of gravity changes while transporting the bag. The agent also needs to consider these factors for safe performance [117].

4.4.2 Perception in Bag Manipulation

For developing a system that can conduct IC, Seita et al. (2021b) [330] produce a sack mask from a top-down depth camera for generating grasping candidates for a dual-arm robot. They also leverage interactive perception for checking if the target bag/sack is grasped robustly – the robot applies a shaking motion to test if the bag will slip or not [330].

Seita et al. (2021a) [327] and Teng et al. (2022) [362] leverage supervised learning to fit a pixel-based goal-condition value heatmap for P&P action primitives to perform a set of skills for bag manipulations. Weng et al. (2021) [387] investigate graph-based key point dynamic learning for predicting the interaction between rigid items and a bag. They adopt two-stage prediction to reduce prediction error. This dynamic model can provide a prior for the perception system to keep track of the important key points on the bag.

In BU, Kirchheim et al. (2008) [187] and Gonnochenko et al. (2021) [117] infer the depth of individual sacks from a depth sensor using detection and segmentation techniques.

4.4.3 Classical and Data-driven Control in Bag Manipulation

Two end-effectors are required to firmly grasp the appropriate point on the opening of the sack [330] or find the handle of the bag to perform IL. Seita et al. (2021b) [327] adopt a combination of heuristics and imitation learning to learn the grasping position on a sack. The heuristic strategy is to grasp the endpoints of the maximum width of the sack mask. They also leverage data augmentation (rotation and translation) on the current and goal images to improve the representation capability of the system. Xu et al. (2022) [395] propose a system called *DextAirity* that uses air-blowing to open a sack while grasping the hem of the sack with bilateral grippers.

Seita et al. (2021a) [327] adopt a *Transporter Network* [411] (Section 2.1.1), a SOTA pixel-based behaviour cloning algorithm that focuses on pick-and-place action primitives, to learn goal-condition policy for sack opening and item insertion tasks in simulation. Teng et al. (2022) [362] add a dense network layer [158] and residual connections [141] in the transporter to enhance feature extraction, and they further examined the system on ILD and BM. Both approaches train the policy network only with successful trials in simulation induced by a scripted demonstration policy.

Kierchheim et al. (2018) [187] and Gonnochenko et al. (2021) [117] employ a roller end-effector to perform sack unloading. Kierchheim et al. (2018) apply a handcrafted policy for effective grasping, while Gonnochenko et al. (2022) adopted DRL to learn the grasping position and orientation in the *Mujoco* simulator.

Seita et al. (2021a) [327] created a benchmark environment *DeformableRaven* built on *PyBullet* [66] that includes modular sack manipulation environments. Nonetheless, there are opportunities to improve the modelling of the sack, and the benchmark does not contain environments involving bags. Seita et al. (2021b) [330] propose using human teleoperation to set a soft upper bound for manipulation tasks in real-world trials. In simulation, Seita et al. (2021a) [327] achieve a 63.3% success rate out of 60 trials for sack opening and a 51.7% success rate for inserting one item in simulation. Teng et al. (2022) [362] achieved a 48.3% success rate out of 60 trials for item-loading and sack-moving in simulation.

5 Inverse Reinforcement Learning for CDO manipulation

As type of data-driven IL algorithms, inverse reinforcement learning is investigated comparatively little in any of the four CDO task families. The major reason is the need for feature engineering in conventional IRL algorithms, and the need for an inner loop of RL optimisation to get an optimal policy for an inferred reward function. Nevertheless, the recent advancement of Adversarial IRL make it possible to apply these techniques to such complex domains. These methods are examined in the standard RL benchmark environments, especially *V-MAIL* [293] shows substantial improvement compared to BC in continuous control settings. Adversarial IRL methods also eliminate the effort needed to engineer rewards for using RL in complex tasks, which is a challenge for knot-tying, multi-step cloth-folding and dressing tasks. In this section, we talk about the fundamentals of conventional IRL methods for better comprehension of the SOTA adversarial IRL approaches.

Inverse reinforcement learning (IRL), also known as inverse optimal control, aims to learn a reward function from the given demonstration policy. This objective is preferable to the one of behaviour cloning when we need to re-optimise a reward in a new environment [101] or to infer the demonstrator's intention. It is also a natural framework for LfO by learning a state-dependent reward function [271].

The general objective of IRL is to find a reward function that makes demonstration policy better than other policies:

$$\mathrm{IRL}(\pi^{demo}, \mathcal{P}) \doteq \underset{r\in\mathbb{R}^{\mathcal{S}\times\mathcal{A}}}{\arg\max} \underset{\tau\sim d^{\pi^{\text{demo}}}}{\mathbb{E}}\Big[\sum_{t=1}^{T} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\Big] - \max_{\pi\in\Pi} \underset{\tau\sim d^{\pi}}{\mathbb{E}}\Big[\sum_{t=1}^{T} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\Big] \quad . \tag{32}$$

Apprenticeship learning [3] is the first IRL method that formulates the objective with feature matching of state-action pairs, where it assumes that the reward is the linear combination of the state-action features. However, feature-matching IRL methods cannot deal with the cases where the reward function is essentially complex and non-linear. Apart from that, the objective of feature matching is ill-posed, as an optimal policy may be achieved by an infinite amount of reward [266] and an infinite number of policies have the same expected feature as the demonstration policy. The *Maximum Margin Principle* [297] and *Maximum Entropy Principle* [421] have been applied to mitigate those ambiguities in the FM-IRL objectives.

Under the framework of MaxEnt-IRL, methods with non-linear reward functions parameterised using NNs, such as *GAN-GCL* [100, 98] and *GAIL* [150], have been proposed to tackle the high dimensional continuous setting with unknown dynamics. In addition to NNs, boosting [19] and Gaussian process [227] have both been adopted, but these methods still require manual feature engineering. NN-based IRL methods can learn a complex reward function through automatic feature learning. Moreover, they alleviate the need to search a policy from scratch for an updated reward function in the inner loop. Since they are closely related to adversarial training of generative models [98, 119], we refer to these as Adversarial IRL methods. They suffer from the same issues that occur in the GAN training. Regularising techniques, such as gradient penalty [122], spectral norm [253], *Mixup* [416] and the *PUGAIL* loss [394], have been adopted to stabilise convergence. Since *GCL* and *GAIL* are on-policy methods, they also suffer from data efficiency. Off-policy methods, such as *DAC* [197] and *SQIL* [299], have been proposed to overcome this issue.

5.1 Feature Matching IRL

In feature matching IRL (FM-IRL) [3], the reward function is parameterised as a linear combination of the features of the state-action pair:

$$r(\boldsymbol{s},\boldsymbol{a};\boldsymbol{w},\phi) = \boldsymbol{w}^{\top}\phi(\boldsymbol{s},\boldsymbol{a}) \quad , \tag{33}$$

where ϕ is the feature extraction function defined on a state-action pair. **Feature expectation** of a policy is defined as:

$$\bar{\phi}(\pi) = \underset{\tau\sim d^{\pi}}{\mathbb{E}}\big[f(\tau)\big] \;\text{ and }\; \bar{\phi}(\pi,\boldsymbol{s}) = \underset{\tau\sim d^{\pi}}{\mathbb{E}}\big[f(\tau)\mid \boldsymbol{s}_1=\boldsymbol{s}\big] \quad , \tag{34}$$

where $f(\tau) = \sum_{t=1}^{T}\gamma^t\phi(\boldsymbol{s}_t,\boldsymbol{a}_t)$. Hence, the value function of a policy of a certain state is:

$$V^{\pi}(\boldsymbol{s}) = \boldsymbol{w}^{\top}\bar{\phi}(\pi,\boldsymbol{s}) \quad . \tag{35}$$

The objective of IRL is to find the parameter $\boldsymbol{w}^*$ that makes the demonstration policy produce a higher reward than any other policies:

$$\text{FM-IRL}(\pi^{demo},\mathcal{P}) \doteq \arg\max_{\boldsymbol{w}} \boldsymbol{w}^{\top}\bar{\phi}(\pi^{\text{demo}}) - \max_{\pi\in\Pi} \boldsymbol{w}^{\top}\bar{\phi}(\pi) \quad . \tag{36}$$

Abbeel and Ng (2004) [3] propose *Apprenticeship Learning* (AL) to find a policy π that has the similar feature expectation as that of the demonstration policy by reformulating Equation 36. The disadvantage of this algorithm is that it is expensive to scale up to complex and high-dimensional settings because it needs to run an underlying RL algorithm after each iteration. Besides, there is an infinite number of reward functions with the same optimal policy, and there is an infinite amount of stochastic policies that can satisfy feature matching property.

Ratliff et al. (2016) propose *Maximum Margin Planning* (MMP) [297] as an improvement on AL that modifies the objective function to add flexibility. One can add a slack term d and a margin function, which measures the dissimilarity between two policies, to account for the suboptimality of the demonstration. However, maximising the margin is arbitrary [272], and accommodating the sub-optimality using slack variables is still a form of heuristic.

Apart from the ambiguity of $\boldsymbol{w}$, feature expectation $\bar{\phi}(\pi)$ can also be ambiguous, i.e., two different stochastic policies can map to the same feature expectation. Suppose a policy π has the same feature expectation as the demonstration policy. We have

$$\int d^{\pi}(\tau) f(\tau) d\tau = \int d^{\pi^{\text{demo}}}(\tau) f(\tau) d\tau \quad , \tag{37}$$

where trajectory distribution d^{π} and $d^{\pi^{\text{demo}}}$ can be different but still make the equality hold.

Ziebart et al. (2010) [421] propose *Maximum Entropy IRL* (MaxEnt-IRL) on FM-IRL that leverages the *Maximum Entropy Principle* [172] to remove this distribution ambiguity by finding a policy that matches the demonstration feature expectation but no other path preferences. The maximum entropy objective is defined as follows:

$$\begin{aligned} \arg\max_{\pi} \quad & \int -d^{\pi}(\tau) \log d^{\pi}(\tau) d\tau \\ \text{s.t.} \quad & \int d^{\pi}(\tau) f(\tau)\, d\tau = \int d^{\pi^{\text{demo}}}(\tau) f(\tau)\, d\tau \\ & \int d^{\pi}(\tau) d\tau = 1 \\ & d^{\pi}(\tau) \geq 0 \quad \forall \tau \quad . \end{aligned} \tag{38}$$

We can generalise the MaxEnt-IRL objective, which can be derived with *Control as Inference* [225]:

$$\begin{aligned} \text{MaxEnt-IRL}(\pi^{demo}, \mathcal{P}, \alpha) \doteq \arg\max_{r} & \mathop{\mathbb{E}}_{\tau \sim d^{\pi^{\text{demo}}}} \Big[\sum_{t=1}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t) \Big] - \\ & \Bigg(\max_{\pi} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho_\pi} \Big[r(\boldsymbol{s}_t, \boldsymbol{a}_t) \Big] + \alpha \mathop{\mathbb{E}}_{\boldsymbol{s}_t \sim \rho_\pi} \Big[\mathcal{H}\big(\pi(\cdot \mid \boldsymbol{s}_t)\big) \Big] \Bigg) \quad . \end{aligned} \tag{39}$$

The solution of the maximum entropy objective in FM-IRL can be obtained by approximating the trajectory distribution with an exponential form on the reward function regulated by a set of hyper-parameters $\boldsymbol{\lambda}$:

$$d^{\pi}(\tau) \approx q(\tau; \boldsymbol{\lambda}) \doteq \frac{1}{Z(\boldsymbol{\lambda})} \exp(\boldsymbol{\lambda}^{\top} f(\tau)) \quad , \tag{40}$$

where $Z(\boldsymbol{\lambda}) = \int \exp(\boldsymbol{\lambda}^{\top} f(\tau)) d\tau$ is the partition function. Shiarlis et al.(2016) [334] improved MaxEnt-FM-RL by also taking failed demonstrations into account.

5.2 Adversarial IRL

FM-IRL methods are developed using a linearly parameterised reward function that depends on a hand-designed feature function on state-action pairs. Also, these methods are hard to apply to high dimensional continuous state-action settings, especially with unknown dynamics, mainly due to the inner loop of the policy training using RL. For learning complex reward functions with formative features, we can parameterise the reward function with a neural network:

$$r(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{w}) \doteq NN(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{w}) \quad . \tag{41}$$

There have been many algorithms proposed under the MaxEnt-IRL framework, such as *GCL* [100], *GAIL* [150], *AIRL* [105], *DAC* [197] and *V-MAIL* [293]. The common trait of these methods is that the reward function is optimised using gradient (Equation 46), and the policy is updated using policy-gradient objectives. So far, there have been no examples of Adversarial IRL methods in CDO domain, but this might the key technology to mitigate the reward engineering obstacle of cloth manipulation tasks while preserving similar computational complexity as their DRL counterparts.

5.2.1 GAN

The training of Adversarial IRL agents is related to the loss function of *Generative Adversarial Networks* (GANs), which is a generative model (Section 3.2.2). Before getting into the details about the Adversarial IRL algorithms, we briefly introduce this model. Goodfellow (2014) propose *Generative Adversarial Networks* (GANs) [119] that generate unseen samples by training a generator G_θ and discriminator D_ϕ. The generator produces samples from a latent space, and the discriminator classifies a sample as real or fake:

$$\text{GAN}(\mathcal{D}) \doteq \min_\theta \max_\phi \mathop{\mathbb{E}}_{x\sim p_\mathcal{D}}\big[\log D_\phi(x)\big] + \mathop{\mathbb{E}}_{x\sim p_\theta}\big[\log(1 - D_\phi(x))\big] \quad , \tag{42}$$

where $D_\phi(x) = 1$ represents x is sampled from the real dataset, and $D_\phi(x) = 0$ represent x is sampled from the generator. This objective is mathematically equivalent to the Jensen-Shannon divergence between real data distribution and the generator distribution while assuming the discriminator has infinite capacity. As the generator produces the sampled data from a latent distribution, the objective can be rewritten as

$$\text{GAN}(\mathcal{D}) \doteq \min_\theta \max_\phi \mathop{\mathbb{E}}_{x\sim p_\mathcal{D}}\big[\log D_\phi(x)\big] + \mathop{\mathbb{E}}_{z\sim p_z}\big[\log(1 - D_\phi(G_\theta(z)))\big] \quad . \tag{43}$$

In practice, we optimise the discriminator parameter ϕ first by fixing θ with gradient ascent using the objective, which is approximated by sampling, then optimises the generator parameter θ with gradient descent on the generated samples. Note that it uses $\min_\theta \; - \mathop{\mathbb{E}}_{x\sim p_\theta} \log D_\phi(x)$ instead of $\min_\theta \; - \mathop{\mathbb{E}}_{x\sim p^\theta}\big[\log(1 - D_\phi(x))\big]$ to optimise the generator for avoiding saturation of the original objective. With a fixed generator distribution p_θ, the optimal discriminator is:

$$D^*(x) = \frac{p_\mathcal{D}(x)}{p_\mathcal{D}(x) + p_\theta(x)} \quad . \tag{44}$$

5.2.2 Adversarial IRL methods

Finn et al. (2016) propose *Guided Cost Learning (GCL)* [100] as the first model-free IRL algorithm that learns the non-linear reward function and a stochastic policy network π_θ at the same time. It is built upon MaxEnt-IRL and designed to apply in high-dimensional complex systems. As the partition function $Z(\boldsymbol{w})$ is expensive to compute, *GCL* uses a sample-based method to estimate the second term of the gradient in Equation 46 using mini-batch stochastic gradient descent. It also uses importance sampling to mitigate the estimation bias. The importance weight for each sampled trajectory is:

$$a(\tau;\boldsymbol{w},\boldsymbol{\theta}) \doteq \frac{p(\tau;\boldsymbol{w})}{d^{\pi_\theta}(\tau)} = \frac{\exp(\sum_{t=1}^{T} r(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{w}))}{\prod_{t=1}^{T}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta})} \quad . \tag{45}$$

Hence, the gradient becomes

$$\nabla_{\boldsymbol{w}} = \mathop{\mathbb{E}}_{\tau^{demo}\sim\mathcal{U}}\left[\nabla_{\boldsymbol{w}} R(\tau^{demo};\boldsymbol{w})\right] - \frac{1}{Z(\boldsymbol{w})}\sum_{\tau\sim\mathcal{D}_{sample}} a(\tau;\boldsymbol{w},\boldsymbol{\theta})\nabla_{\boldsymbol{w}} R(\tau;\boldsymbol{w}) \quad , \tag{46}$$

where $Z(\boldsymbol{w}) = \sum_{\tau\sim\mathcal{D}_{sample}} a(\tau;\boldsymbol{w},\boldsymbol{\theta})$. To improve the policy under the updated reward function, *GCL* adopt the policy-gradient of *Guided Policy Search under Unknown Dynamics* [226] that incorporates the MaxEnt-RL objective with KL-divergence constraint on the trajectory based on local linear models. Original *GCL* is developed based on value-based state settings, so it faces challenges in pixel-based POMDP settings, in particular overfitting problems when using stronger networks.

To avoid overfitting, a convex regularisation function ψ is often applied to the reward function. Following from the MaxEnt-IRL objective (Equation 39), the objective of *ψ-regularised MaxEnt-IRL* becomes:

$$\begin{aligned}\text{MaxEnt-IRL}_\psi(\pi^{demo},\mathcal{P},\alpha) \doteq \arg\max_r -\psi(r) + \mathop{\mathbb{E}}_{\tau\sim d^{\pi^{\text{demo}}}}\left[\sum_{t=1}^{T} r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right] - \\ \left(\max_\pi \sum_{t=1}^{T}\mathop{\mathbb{E}}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\rho_\pi}\left[r(\boldsymbol{s}_t,\boldsymbol{a}_t)\right] + \alpha \mathop{\mathbb{E}}_{\boldsymbol{s}_t\sim\rho_\pi}\left[\mathcal{H}(\pi(\cdot\mid\boldsymbol{s}_t))\right]\right) \quad .\end{aligned} \tag{47}$$

Ho et al. (2016) [150] propose *Generative Adversarial Imitation Learning (GAIL)* [1] by claiming that the ψ-regularised MaxEnt-IRL objective (Equation 47) is a dual problem of a *ψ-regularised maximum entropy occupancy measure matching* problem:

$$\begin{aligned}\text{RL}\circ\text{MaxEnt-IRL}_\psi(\pi^{demo},\mathcal{P},\alpha) \equiv \\ \arg\max_\pi \psi^*\left(D\left[\rho_\pi||\rho_{\pi^{demo}}\right]\right) + \alpha\times\mathop{\mathbb{E}}_{\boldsymbol{s}_t\sim\rho_\pi}\left[\mathcal{H}(\pi(\cdot\mid\boldsymbol{s}_t;\theta))\right] \quad ,\end{aligned} \tag{48}$$

where $D[\cdot||\cdot]$ represents the divergence between two distributions. *GAIL* chooses the ψ-regulariser as:

$$\psi_{GA}(r) \doteq \begin{cases}\mathop{\mathbb{E}}_{\rho_{\pi^{demo}}}\left[g(-r(\boldsymbol{s},\boldsymbol{a}))\right] & \text{if } r>0 \\ \infty & \text{otherwise}\end{cases} \quad \text{and} \quad g(x) = \begin{cases}-x-\log(1-e^x) & \text{if } x>0 \\ \infty & \text{otherwise}\end{cases} \quad . \tag{49}$$

GAIL's policy is optimised using entropy-regularised policy gradient with state-action value and *TRPO* constraint, where the gradient is accumulated by the reward inferred from the discriminator. The reward function used for obtaining the state-action value is defined as follows:

$$r(\boldsymbol{s},\boldsymbol{a}) = -\log\left(1-D(\boldsymbol{s},\boldsymbol{a})\right) \quad , \tag{50}$$

where $D(\boldsymbol{s},\boldsymbol{a})$ indicates the possibility of the state-action occurrence in the demonstration trajectory. Hence, the optimal value of the regulariser is:

$$\psi^*_{GA}(D[\rho_\pi||\rho_{\pi^{\text{demo}}}]) = \max_{D\in(0,1)^{\mathcal{S},\mathcal{A}}}\mathop{\mathbb{E}}_{\rho_{\pi^{demo}}}[\log(D(\boldsymbol{s},\boldsymbol{a})] + \mathop{\mathbb{E}}_{\rho_\pi}[\log(1-D(\boldsymbol{s},\boldsymbol{a})] \quad , \tag{51}$$

[1]In the original paper, the algorithm is derived with cost function in the range $(-\infty,0]$. Here, we give the one derived with the reward function for consistency, so the reward function in the range $[0,\infty)$. And, we always want $D(\tau)$ or $D(s,a)$ indicates the possibility of demonstration policy.

which resembles the GAN's original objective. Essentially, *GAIL* minimises the Jensen-Shannon divergence on the occupancy measure of the two policies using this regulariser:

$$GAIL(\pi^{demo}, \mathcal{M}, \alpha) = \arg\max_{\pi} -\mathrm{JS}[\rho_\pi || \rho_{\pi^{demo}}] + \alpha \times \mathop{\mathbb{E}}_{\boldsymbol{s}_t \sim \rho_\pi} \Big[\mathcal{H}(\pi(\cdot \mid \boldsymbol{s}_t; \theta)) \Big] \ . \tag{52}$$

GAIL is an on-policy algorithm, so it is not data-efficient. It often suffers from unstable training caused by similar factors affecting training of GANs [185]. *GCL* and *GAIL* are model-free IRL algorithms that can apply to complex environments without feature engineering. They both learn the reward function and the policy at the same time. The difference is that in *GAIL*, the reward function is indirectly optimised by optimising the discriminator with the occupancy measure matching objective 48, while the one in *GCL* is optimised with the MaxEnt-IRL objective (Equation 39).

Finn et al. (2016) [98] propose *GAN-GCL* that builds the bridge between *GCL* and *GAIL*. The reward function is learned through discriminator training similar to GAIL, where we do not have to calculate the importance weight like in *GCL*. At the same time, the policy π_θ can be optimised similarly to *GCL*. Following from GAN's optimal discriminator representation (Equation 44) and Boltzmann distribution of optimal trajectory as the approximation of the distribution of demonstration trajectory, *GAN-GCL* represent the discriminator w.r.t. the parameterised reward function $r_{\boldsymbol{w}}$:

$$D_{\boldsymbol{w}}(\tau) \doteq \frac{p(\tau)\frac{1}{Z(\boldsymbol{w})}\exp\big(R_{\boldsymbol{w}}(\tau)\big)}{p(\tau)\frac{1}{Z(\boldsymbol{w})}\exp\big(R_{\boldsymbol{w}}(\tau)\big) + p_\theta(\tau)} = \frac{\frac{1}{Z(\boldsymbol{w})}\exp\big(R_{\boldsymbol{w}}(\tau)\big)}{\frac{1}{Z(\boldsymbol{w})}\exp\big(R_{\boldsymbol{w}}(\tau)\big) + \prod_{t=1}^{T}\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t)}, \tag{53}$$

where p_θ is the sampling trajectory induced by policy π_θ.

Fu et al. (2018) [105] argue that policy-invariant reward [266] is not robust to changing the dynamic, and it attempt to learn a reward function which is decoupled from the domain dynamic. Inspired by *GAN-GCL*, they propose *Adversarial Inverse Reinforcement Learning (AIRL)* that trains a discriminator expressed by transformed reward function on occupancy measurement distribution like *GAIL*:

$$D(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi}) = \frac{\exp\big(r'(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi})\big)}{\exp\big(r'(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi})\big) + \pi(\boldsymbol{a}|\boldsymbol{s}; \theta)} \ , \tag{54}$$

where they use policy-invariant reward transformation [266] to define $r'(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi}) = r(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{w}) + \gamma h(\boldsymbol{s}'; \boldsymbol{\phi}) - h(\boldsymbol{s}; \boldsymbol{\phi})$. In order to make reward learning independent of the dynamic, *AIRL* makes $r_{\boldsymbol{w}}$ solely dependent on the state instead of the state-action pair. The reward function that is used to represent the discriminator becomes:

$$r'(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi}) = r(\boldsymbol{s}; \boldsymbol{w}) + \gamma h(\boldsymbol{s}'; \boldsymbol{\phi}) - h(\boldsymbol{s}; \boldsymbol{\phi}), \tag{55}$$

and the reward that is used to update the policy π_θ in *AIRL* is:

$$r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi}) = \log D(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi}) - \log\big(1 - D(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'; \boldsymbol{w}, \boldsymbol{\phi})\big) \ . \tag{56}$$

Kostrikov et al. (2019) introduce the *Discriminator Actor Critic (DAC)* [197] which extends Adversarial IRL to off-policy learning to improve sample efficiency. Moreover, it resolves reward biases in the reward functions defined in *GAIL* and *AIRL* using the absorbing reward function. It shows consistently better performance than *GAIL* and *AIRL* across different domains. After sampling trajectories using an updated policy, the transition of the absorbing states is manually appended to the end of the samples. The discriminator is defined and optimised similarly to *AIRL*, but negative samples are sampled from the replay buffer that stores all the trajectories induced by previous policies. In *DAC*, they adopt *TD3*, instead of *TRPO* like other Adversarial-IRL methods, to optimise the policy π_θ because it provides a good trade-off between sample complexity and implementation for practical usage. Blondé and Kalousis (2019) [38] propose another off-policy variants of *GAIL* called *Sample-efficient Adversarial Mimic*.

Rafailov et al. (2021) [293] present *Variational Model-based Adversarial Imitation Learning (V-MAIL)*, a model that attempts to overcome high sample complexity, non-stationary nature of the learned reward function, and poor representation problems of above-mentioned Adversarial IRL methods. *V-MAIL* shows significant improvement both in data efficiency and final performance compared to model-free adversarial IRL methods by learning dynamic model $\hat{\mathcal{P}}_{\vartheta}$, policy network π_{θ} and discriminator $D_{\boldsymbol{w}}$ on latent representation. While the discriminator is updated using the *GAIL* objective, it maximises the value function of the sample state by rolling out the future states using the learned policy and latent dynamic model.

In Adversarial IRL, the formulation of reward function from the discriminator output becomes an essential factor for the training of a robust policy. Training with different reward functions is associated with minimising different divergences between the marginal state-action distribution of the demonstration and trained policy [271]. Orsini et al. (2021) [271] provide a careful implementation and comparison of the mainstream model-free Adversarial IRL methods. They found out that regularisation techniques such as dropout [339] and weight decay [136] have similar regularisation effect as gradient penalty [122] that is proposed to stabilise GAN training. They also experimentally show that learning from artificial demonstrations is worse than human demonstration. One can also learn a sparse-reward classifier to indicate if the agent reaches its goal or not [337]. Furthermore, instead of learning the reward function, one can also hand-design a sparse reward function based on the final state of the demonstration trajectory [299].

6 Discussion and Future

Cloth-like deformable object manipulation is a very active research area in robotics. Other than the four major task families we discussed in this review, there are also some other tasks in the literature covered in this review, including cloth hanging [244], bed making [329], cloth ironing [229], rope insertion [383] and suturing [168].

Grasping is the first challenge in all CDO manipulation domains. Robust grasping in CDO refers to the accurate grasping of the target point without misgrasping or grasping the underlying layer of the CDO. Besides, robust grasping demands persistent holding of the target point without damaging the CDO while performing a certain subtask. A robust RMS should be able to recognise the failing states and recover from these failures [241, 349]. Most of the failures in cloth-shaping literature are due to the deficiency of robust grasping. The main reason the simulation-trained system fails is that none of the simulators provides accurate collision and friction modelling between the gripper and CDOs. Robust grasping has largely been unexplored in the dressing domain.

Apart from robust grasping, the success and efficiency of CDO manipulation highly rely on effective grasping. This means that the system should target an effective point to grasp to accomplish a certain skill or a subtask. This usually introduces inductive biases to individual task domains. The corners and the contour of the articles are crucial for cloth-shaping tasks. Wakamatsu et al. (2004, 2006a, 2006b) [379, 378, 380] suggest a set of grasping strategies for different topological scenarios for knot tying and untying tasks to accomplish a certain move. In bag manipulation, the systems focus on the opening contour and the handles. However, dressing literature often ignores this problem by letting the agent pregrasp an assigned key point before executing a task. Grasping strategy for dressing tasks is an important uninvestigated research area for achieving a fully automated dressing agent. The grasping strategy is the first type of inductive bias in these four domains. A general agent should be able to flexibly adjust its grasping strategy based on the article type and the goal of the task. This can be delegated to individual motor skills.

The application of reinforcement learning has been heavily investigated in the cloth-shaping and dressing domains. Meanwhile, there are fewer DRL applications in dressing and none in bag and rope manipulation. Reward shaping is one of the open problems of applying RL in CDO. Although there have been several attempts [155, 60, 358], designing a simple and effective dense reward function for cloth folding and dressing tasks is still challenging. A sparse reward function can solve this, but detecting the success of a particular subtask and the full task automatically is equally difficult. Moreover, we are aware of no literature on solving knot tying/untying using RL, probably because of the difficulty in defining dense reward functions in this domain. A possible solution is to take reference from energy function [336, 205] from knot theory literature, but this can only solve untying problems using RL.

The second inductive bias is the difference in the reward functions between the 4 task families. A promising solution to avoid reward shaping in CDO is to use imitation learning methods. Imitation learning approaches have been investigated in detail in all four domains. However, these applications are mostly based on behaviour cloning (BC) and learning-from-observation (LfO) methods. We are unaware of approaches based on inverse reinforcement learning (IRL), a type of data-driven imitation learning method, in CDO manipulation literature. The recent advancement of Adversarial IRL shows substantial improvements compared to BC baselines in continuous control settings [293]. This could be the key technology to bypass the complex reward engineering in CDO domain while persevering similar data-efficiency as DRL methods.

Conventional IRL methods are inefficient mainly due to the inner loop of RL optimisation. Adversarial IRL approaches have in the past five years improved the training efficiency by learning the reward function and policy at the same time. It will be interesting to see the performance of adversarial IRL methods in the CDO domain. Among the four domains, only some of the application on rope manipulation have utilised LfO approaches with the assistance of topological perception and motion planning approaches. LfO with IRL is probably the closest analogy to how humans attempt to imitate the demonstrator's intention. It will be interesting to see the application of such approaches in the CDO domain.

The differences in the intermediate representations and perception systems among the four domains account for the third and the fourth inductive biases. In cloth-shaping, most systems keep track of the counter and corners of the cloth, while some attempt to reconstruct the mesh representation. Most of the knot tying/untying systems leverage topological representation of the rope, while some keep track of the individual points on the rope. In bag manipulation, RMSs are mainly interested in the openings and handles as well as the size and shape of the bag. In the dressing domain, we are mainly interested in the relationship between the article and the body. We also want to keep track of human motion for safe and reactive manipulation. Similar to the grasping strategy, the variation of perception systems can be delegated to individual motor skills. Many approaches also attempt to learn a latent representation in self-supervised and unsupervised manners for generalisation. Representation-learning community in DRL and DIL can take inspiration from the challenges of perception in these four domains to propose more general and effective representation learning algorithm. Moreover, applications of transfer learning, multi-task and continual learning in the CDO domain are interesting directions to explore.

To develop a robust RL skill controller, we cannot avoid the exploration component because the agent needs to encounter different scenarios to learn to achieve goal states from them. SOTA RL exploration strategies are barely applied in any of these domains. The major obstacle for the exploration is CDO's large state-action space and complexity of the state-action dynamic. It will be interesting to see if the SOTA exploration strategies can improve the data efficiency of RL in CDO domains. Skill-level environments of CDO could also be a good development benchmark for the progression of DRL and DIL, where CDO set challenges to exploration and state estimation for these data-driven methods due to its complex state-action dynamic and sever self-occlusions.

Multi-modal learning in DRL has not been explored much in mainstream research. In reality, humans are highly reliant on haptic sensors to control the force and infer the material property of objects in addition to vision. In the CDO domain, we can tie a knot without looking at the rope, and we depend on the haptic sensor to dress and even help others to dress garments. Furthermore, we can roughly estimate how much force we can exert on a certain object. Building an observational world model that incorporates both vision and haptic signals is an interesting research direction for robotics development.

Velocity/acceleration control is still a challenging problem in cloth-shaping, even in simulation. Theoretically, velocity/acceleration control can be more efficient than P&P action primitives because fling action can take advantage of the inertia of the cloth [126] for flattening. It is more fine-grained control for smoothing the surface of the cloth at the end of folding and flattening. Methods using the analytical model can only perform velocity control in a narrow range of fixed configurations. For more robust dynamic control, the imitation learning and reinforcement learning approaches suffer from data efficiency caused by the enormous state-action space of the domain. For such fine-grained control, the delay between the perception and control can devastate the system's performance in physical trials. We will need to develop novel algorithms based on *Action-concurrent Continuous-Time MDP* [393].

There exist two simulation benchmark environments in the CDO domain; *SoftGym* [234] includes a variety of cloth-shaping environments, while *DeformableRaven* [327] also offers bag-manipulation configurations. However, neither environment can model the interaction between the cloth and the gripper; they adopt anchoring to attach the cloth to the gripper. Also, the modelling of the rope and the hem of the sack is based on a sequence of beads in *DeformableRaven*, which introduces a significant reality gap. Robotic application on bags with handles are rare compared to sacks in bag manipulation. Most of the dressing simulation is conducted in the *Nvidia PhysX* [2] simulator, but there is no benchmark environment for this task family. There is no standard simulation and benchmark environment for rope insertion, knot tying and knot untying tasks, although the rendering of the ropes has been done using *Blender* and *Unity with Obi*. Furthermore, precise and tight knot-tying has not yet been studied in sufficient depth, and we are unaware of any robotic applications in assistive undressing and self-undressing tasks. Improving the simulation and creating more skill-related benchmark environments in the CDO domain could accelerate the progress in this field.

6.1 Summary

This review has covered the state of the art developments in four tasks families in CDO manipulation, including cloth-shaping, rope manipulation, bag manipulation and dressing. Most of the systems focus on skill developments, where each task domain is beginning to adopt data-driven approaches, such as deep imitation learning and deep reinforcement learning in order to achieve more robust and general skill controllers. Attempts at solving long-horizon multi-step tasks that involve multiple articles and other items are rare in this field. It will be beneficial to build benchmark environments for tasks like doing laundry [133, 241] and a full set of assistive/self-dressing/undressing tasks. We expect that this will require involvement of hierarchical control methods based on the frameworks of *options* and *semi-MDP* [25].

This review identifies four types of inductive biases that occurs in the four task families, which are the differences in grasping strategies, reward engineering, intermediate representations and perception systems. We also outline the recent advancement and tools of robotics, DIL and DRL that can be employed in CDO manipulations, along with their challenges so that readers will be aware of these obstacles while applying them. In the end, we summarised and identified the future work in CDO manipulation. further develop CDO manipulation.

References

[1] I-dress: Assistive interactive robotic system for support in dressing. `https://i-dress-project.iri.upc.edu/`.

[2] Nvidia physx 4.5 and 5.0 sdk. `https://developer.nvidia.com/physx-sdk`, Aug 2022.

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

[4] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[5] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[6] Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

[7] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. *Advances in neural information processing systems*, 32, 2019.

[8] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR, 2017.

[9] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[10] Solvi Arnold, Daisuke Tanaka, and Kimitoshi Yamazaki. Cloth manipulation planning on basis of mesh representations with incomplete domain knowledge and voxel-to-mesh estimation. *arXiv preprint arXiv:2103.08137*, 2021.

[11] Ted Ashton, Jason Cantarella, Michael Piatek, and Eric J Rawdon. Knot tightening by constrained gradient descent. *Experimental Mathematics*, 20(1):57–90, 2011.

[12] Alexandre Attia and Sharone Dayan. Global overview of imitation learning. *arXiv preprint arXiv:1801.06503*, 2018.

[13] Arthur Aubret, Laetitia Matignon, and Salima Hassas. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

[14] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[15] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE, 2018.

[16] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *International Conference on Learning Representations*, 2018.

[17] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

[18] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

[19] J Bagnell, Joel Chestnutt, David Bradley, and Nathan Ratliff. Boosting structured prediction for imitation learning. *Advances in Neural Information Processing Systems*, 19, 2006.

[20] J Andrew Bagnell. An invitation to imitation. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 2015.

[21] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[22] Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. End-to-end differentiable adversarial imitation learning. In *International Conference on Machine Learning*, pages 390–399. PMLR, 2017.

[23] Justyna Baranska, Sylwester Przybyl, and Piotr Pieranski. Curvature and torsion of the tight closed trefoil knot. *The European Physical Journal B*, 66(4):547–556, 2008.

[24] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[25] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1):41–77, 2003.

[26] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[27] Matthew Bell. *Flexible object manipulation*. Dartmouth College, 2010.

[28] Matthew P Bell, Weifu Wang, Jordan Kunzika, and Devin Balkcom. Knot-tying with four-piece fixtures. *The International Journal of Robotics Research*, 33(11):1481–1489, 2014.

[29] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[30] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[31] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[32] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[33] Jur van den Berg, Stephen Miller, Ken Goldberg, and Pieter Abbeel. Gravity-based robotic cloth folding. In *Algorithmic Foundations of Robotics IX*, pages 409–424. Springer, 2010.

[34] Christian Bersch, Benjamin Pitzer, and Sören Kammel. Bimanual robotic cloth manipulation for laundry folding. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1413–1419. IEEE, 2011.

[35] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

[36] Dimitri Bertsekas. *Reinforcement learning and optimal control.* Athena Scientific, 2019.

[37] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

[38] Lionel Blondé and Alexandros Kalousis. Sample-efficient imitation learning via generative adversarial nets. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3138–3148. PMLR, 2019.

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

[40] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[41] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[42] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[43] Andrea Calanca, Riccardo Muradore, and Paolo Fiorini. A review of algorithms for compliant control of stiff and fixed-compliance robots. *IEEE/ASME transactions on mechatronics*, 21(2):613–624, 2015.

[44] Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. Sparse multi-task reinforcement learning. *Advances in neural information processing systems*, 27, 2014.

[45] Sylvain Calinon, Florent D'halluin, Eric L Sauser, Darwin G Caldwell, and Aude G Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine*, 17(2):44–54, 2010.

[46] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control.* Springer science & business media, 2013.

[47] Mathias Carlen, Ben Laurie, John H Maddocks, and Jana Smutny. Biarcs, global radius of curvature, and the computation of ideal knot shapes. In *Physical and Numerical Models in Knot Theory: Including Applications to the Life Sciences*, pages 75–108. World Scientific, 2005.

[48] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[49] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional vrnns for video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7608–7617, 2019.

[50] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

[51] Greg Chance, Antonella Camilleri, Benjamin Winstone, Praminda Caleb-Solly, and Sanja Dogramadzi. An assistive robot to support dressing-strategies for planning and error handling. In *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 774–780. IEEE, 2016.

[52] Greg Chance, Aleksandar Jevtić, Praminda Caleb-Solly, and Sanja Dogramadzi. A quantitative analysis of dressing dynamics for robotic dressing assistance. *Frontiers in Robotics and AI*, 4:13, 2017.

[53] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pages 1430–1440. PMLR, 2021.

[54] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3381–3388. IEEE, 2017.

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

[56] Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009.

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

[58] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28:2980–2988, 2015.

[59] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

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

[61] Alexander Clegg, Wenhao Yu, Zackory Erickson, Jie Tan, C Karen Liu, and Greg Turk. Learning to navigate cloth using haptics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2799–2805. IEEE, 2017.

[62] Alexander Clegg, Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018.

[63] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[64] Adria Colomé, Antoni Planells, and Carme Torras. A friction-model-based framework for reinforcement learning of robotic tasks in non-rigid environments. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 5649–5654. IEEE, 2015.

[65] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

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

[67] Richard H Crowell and Ralph Hartzler Fox. *Introduction to knot theory*, volume 57. Springer Science & Business Media, 2012.

[68] Marco Cusumano-Towner, Arjun Singh, Stephen Miller, James F O'Brien, and Pieter Abbeel. Bringing clothing into desired configurations with limited perception. In *2011 IEEE international conference on robotics and automation*, pages 3893–3900. IEEE, 2011.

[69] Neha Das, Sarah Bechtle, Todor Davchev, Dinesh Jayaraman, Akshara Rai, and Franziska Meier. Model-based inverse reinforcement learning from visual demonstrations. In *Conference on Robot Learning*, pages 1930–1942. PMLR, 2021.

[70] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[71] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[72] Miha Deniša, Andrej Gams, Aleš Ude, and Tadej Petrič. Learning compliant movement primitives through demonstration and statistical generalization. *IEEE/ASME transactions on mechatronics*, 21(5):2581–2594, 2015.

[73] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International conference on machine learning*, pages 1174–1183. PMLR, 2018.

[74] Renaud Detry, Carl Henrik Ek, Marianna Madry, and Danica Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *2013 IEEE International Conference on Robotics and Automation*, pages 601–608. IEEE, 2013.

[75] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

[76] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[77] Andreas Doumanoglou, Andreas Kargakos, Tae-Kyun Kim, and Sotiris Malassiotis. Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 987–993. IEEE, 2014a.

[78] Andreas Doumanoglou, Tae-Kyun Kim, Xiaowei Zhao, and Sotiris Malassiotis. Active random forests: An application to autonomous unfolding of clothes. In *European Conference on Computer Vision*, pages 644–658. Springer, 2014b.

[79] Andreas Doumanoglou, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimir Petrik, Andreas Kargakos, Libor Wagner, Václav Hlaváč, Tae-Kyun Kim, and Sotiris Malassiotis. Folding clothes autonomously: A complete pipeline. *IEEE Transactions on Robotics*, 32(6):1461–1478, 2016.

[80] Clifford H Dowker and Morwen B Thistlethwaite. Classification of knot projections. *Topology and its Applications*, 16(1):19–31, 1983.

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

[82] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *CoRR*, abs/1901.10995, 2019.

[83] Stefan Eickeler, Andreas Kosmala, and Gerhard Rigoll. Hidden markov model based continuous online gesture recognition. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, volume 2, pages 1206–1208. IEEE, 1998.

[84] Zackory Erickson, Alexander Clegg, Wenhao Yu, Greg Turk, C Karen Liu, and Charles C Kemp. What does the person feel? learning to infer applied forces during robot-assisted dressing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6058–6065. IEEE, 2017.

[85] Zackory Erickson, Henry M Clever, Greg Turk, C Karen Liu, and Charles C Kemp. Deep haptic model predictive control for robot-assisted dressing. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4437–4444. IEEE, 2018.

[86] Zackory Erickson, Maggie Collier, Ariel Kapusta, and Charles C Kemp. Tracking human pose during robot-assisted dressing using single-axis capacitive proximity sensing. *IEEE Robotics and Automation Letters*, 3(3):2245–2252, 2018.

[87] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[88] Marco Ewerton, Guilherme Maeda, Gerrit Kollegger, Josef Wiemeyer, and Jan Peters. Incremental imitation learning of context-dependent motor skills. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 351–358. IEEE, 2016.

[89] Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *International Conference on Learning Representations*, 2021.

[90] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[91] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021.

[92] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[93] Zheming Fan, Wanpeng Shao, Toyohiro Hayashi, and Takeshi Ohashi. Untying cable by combining 3d deep neural network with deep reinforcement learning. *Advanced Robotics*, pages 1–15, 2022.

[94] Gerald Farin. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.

[95] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*, pages 283–321. Springer, 2012.

[96] Paul Fearnhead and Zhen Liu. On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):589–605, 2007.

[97] Thomas Fink and Yong Mao. *The 85 ways to tie a tie: the science and aesthetics of tie knots*. Broadway, 2000.

[98] Chelsea Finn, Paul F. Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852, 2016.

[99] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[100] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[101] Chelsea Finn, Tianhe Yu, Justin Fu, Pieter Abbeel, and Sergey Levine. Generalizing skills with semi-supervised reinforcement learning. *CoRR*, abs/1612.00429, 2016.

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

[103] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2017.

[104] Emily Fox, Michael Jordan, Erik Sudderth, and Alan Willsky. Sharing features among dynamical systems with beta processes. *Advances in neural information processing systems*, 22, 2009.

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

[106] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[107] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. Rotors—a modular gazebo mav simulator framework. In *Robot operating system (ROS)*, pages 595–625. Springer, 2016.

[108] Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2188–2202, 2011.

[109] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[110] Yang Gao, Lisa Anne Hendricks, Katherine J Kuchenbecker, and Trevor Darrell. Deep learning for tactile understanding from visual and haptic data. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 536–543. IEEE, 2016.

[111] Yixing Gao, Hyung Jin Chang, and Yiannis Demiris. User modelling for personalised dressing assistance by humanoid robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1840–1845. IEEE, 2015.

[112] Yixing Gao, Hyung Jin Chang, and Yiannis Demiris. Iterative path optimisation for personalised dressing assistance using vision and force information. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4398–4403. IEEE, 2016.

[113] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[114] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. In *International Conference on Learning Representations*, 2018.

[115] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Scholkopf. From variational to deterministic autoencoders. In *International Conference on Learning Representations*, 2020.

[116] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[117] Aleksei Gonnochenko, Aleksandr Semochkin, Dmitry Egorov, Dmitrii Statovoy, Seyedhassan Zabihifar, Aleksey Postnikov, Elena Seliverstova, Ali Zaidi, Jayson Stemmler, and Kevin Limkrailassiri. Coinbot: Intelligent robotic coin bag manipulation using artificial brain. In *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, pages 67–74, 2021.

[118] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

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

[120] Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeffrey Ichnowski, Ashwin Balakrishna, Minho Hwang, Vainavi Viswanath, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. Untangling dense knots by learning task-relevant keypoints. *arXiv preprint arXiv:2011.04999*, 2020.

[121] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pages 2829–2838. PMLR, 2016.

[122] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[123] Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Rémi Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

[124] Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pages 3875–3886. PMLR, 2020.

[125] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[126] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*, pages 24–33. PMLR, 2022.

[127] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

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

[129] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[130] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[131] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

[132] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *CoRR*, abs/1502.02259, 2015.

[133] Kyoko Hamajima. Planning strategy for task untangling laundry-isolating clothes from a washed mass. *Robotics Mechatron.*, 10:244–251, 1998.

[134] Kyoko Hamajima and Masayoshi Kakikura. Planning strategy for task of unfolding clothes. *Robotics and Autonomous Systems*, 32(2-3):145–152, 2000.

[135] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.

[136] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, 1, 1988.

[137] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[138] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

[139] He He, Jason Eisner, and Hal Daume. Imitation learning by coaching. *Advances in neural information processing systems*, 25, 2012.

[140] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[141] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[142] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020.

[143] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[144] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

[145] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[146] Julius Hietala, David Blanco-Mulero, Gokhan Alcan, and Ville Kyrki. Closing the sim2real gap in dynamic cloth manipulation. *arXiv preprint arXiv:2109.04771*, 2021.

[147] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[148] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

[149] Edmond SL Ho and Taku Komura. Character motion synthesis by topology coordinates. In *Computer Graphics Forum*, volume 28, pages 299–308. Wiley Online Library, 2009.

[150] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[151] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[152] Heiko Hoffmann, Peter Pastor, Dae-Hyung Park, and Stefan Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation*, pages 2587–2592. IEEE, 2009.

[153] John E Hopcroft, Joseph K Kearney, and Dean B Krafft. A case study of flexible object manipulation. *The International Journal of Robotics Research*, 10(1):41–50, 1991.

[154] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Visuospatial foresight for multi-step, multi-task fabric manipulation. *CoRR*, abs/2003.09044, 2020.

[155] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Visuospatial foresight for physical sequential fabric manipulation. *Autonomous Robots*, 46(1):175–199, 2022a.

[156] Ryan Hoque, Kaushik Shivakumar, Shrey Aeron, Gabriel Deza, Aditya Ganapathi, Adrian Wong, Johnny Lee, Andy Zeng, Vincent Vanhoucke, and Ken Goldberg. Learning to fold real garments with one arm: A case study in cloud-based robotics research. *CoRR*, abs/2204.10297, 2022b.

[157] Ronald A Howard. Dynamic programming and markov processes. 1960.

[158] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

[159] Milana Huang, Robert P Grzeszczuk, and Louis H Kauffman. Untangling knots by stochastic energy optimization. In *Proceedings of Seventh Annual IEEE Visualization'96*, pages 279–286. IEEE, 1996.

[160] Sandy H Huang, Jia Pan, George Mulcaire, and Pieter Abbeel. Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 878–885. IEEE, 2015.

[161] Zixuan Huang, Xingyu Lin, and David Held. Mesh-based dynamics with occlusion reasoning for cloth manipulation. *arXiv preprint arXiv:2206.02881*, 2022.

[162] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[163] Auke Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems*, 15, 2002.

[164] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[165] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[166] Masayuki Inaba and Hirochika Inoue. Hand eye coordination in rope handling. *Journal of the Robotics Society of Japan*, 3(6):538–547, 1985.

[167] Masayuki Inaba and Hirochika Inoue. Rope handling by a robot with visual feedback. *Advanced robotics*, 2(1):39–54, 1987.

[168] Russell C Jackson, Viraj Desai, Jean P Castillo, and M Cenk Çavuşoğlu. Needle-tissue interaction force state estimation for robotic surgical suturing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3659–3664. IEEE, 2016.

[169] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017.

[170] Stephen James, Andrew J Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *Conference on Robot Learning*, pages 334–343. PMLR, 2017.

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

[172] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[173] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2017.

[174] Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354, 1992.

[175] Ravi Prakash Joshi, Nishanth Koganti, and Tomohiro Shibata. A framework for robotic clothing assistance by imitation learning. *Advanced Robotics*, 33(22):1156–1174, 2019.

[176] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer, 1993.

[177] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[178] Nal Kalchbrenner, Aäron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, pages 1771–1779. PMLR, 2017.

[179] Rudolf E. Kálmán. When is a linear control system optimal. *Journal of Basic Engineering*, 86:51–60, 1963.

[180] Manabu Kaneko and Masayoshi Kakikura. Planning strategy for putting away laundry-isolating and unfolding task. In *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century.(Cat. No. 01TH8560)*, pages 429–434. IEEE, 2001.

[181] Ariel Kapusta, Wenhao Yu, Tapomayukh Bhattacharjee, C Karen Liu, Greg Turk, and Charles C Kemp. Data-driven haptic perception for robot-assisted dressing. In *2016 25th IEEE international symposium on robot and human interactive communication (RO-MAN)*, pages 451–458. IEEE, 2016.

[182] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE conference on computational intelligence and games (CIG)*, pages 1–8. IEEE, 2016.

[183] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[184] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. *Advances in Neural Information Processing Systems*, 26, 2013.

[185] Kee-Eung Kim and Hyun Soo Park. Imitation learning via kernel mean embedding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[186] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

[188] Yasuyo Kita, Toshio Ueshiba, Ee Sian Neo, and Nobuyuki Kita. Clothes state recognition using 3d observed data. In *2009 IEEE International Conference on Robotics and Automation*, pages 1220–1225. IEEE, 2009.

[189] Yasuyo Kita, Toshio Ueshiba, Ee Sian Neo, and Nobuyuki Kita. A method for handling a specific part of clothing by dual arms. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4180–4185. IEEE, 2009.

[190] Steven D Klee, Beatriz Quintino Ferreira, Rui Silva, Joao Paulo Costeira, Francisco S Melo, and Manuela Veloso. Personalized assistance for dressing users. In *International Conference on Social Robotics*, pages 359–369. Springer, 2015.

[191] Nishanth Koganti, Jimson Gelbolingo Ngeo, Tamei Tomoya, Kazushi Ikeda, and Tomohiro Shibata. Cloth dynamics modeling in latent spaces and its application to robotic clothing assistance. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3464–3469. IEEE, 2015.

[192] Nishanth Koganti, Tomoya Tamei, Kazushi Ikeda, and Tomohiro Shibata. Bayesian nonparametric learning of cloth models for real-time state estimation. *IEEE Transactions on Robotics*, 33(4):916–931, 2017.

[193] Nishanth Koganti, Tomoya Tamei, Takamitsu Matsubara, and Tomohiro Shibata. Estimation of human cloth topological relationship using depth sensor for robotic clothing assistance. In *Proceedings of Conference on Advances In Robotics*, pages 1–6, 2013.

[194] Nishanth Koganti, Tomoya Tamei, Takamitsu Matsubara, and Tomohiro Shibata. Real-time estimation of human-cloth topological relationship using depth sensor for robotic clothing assistance. In *The 23rd IEEE international symposium on robot and human interactive communication*, pages 124–129. IEEE, 2014.

[195] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE, 2018.

[196] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[197] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2019.

[198] K. Kristinsson and G.A. Dumont. System identification and control using genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1033–1046, 1992.

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

[200] Oliver Kroemer, Emre Ugur, Erhan Oztop, and Jan Peters. A kernel-based approach to direct action perception. In *2012 IEEE international Conference on Robotics and Automation*, pages 2605–2610. IEEE, 2012.

[201] Oliver Kroemer, Herke Van Hoof, Gerhard Neumann, and Jan Peters. Learning to predict phases of manipulation tasks as hidden states. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4009–4014. IEEE, 2014.

[202] Shunsuke Kudoh, Tomoyuki Gomi, Ryota Katano, Tetsuo Tomizawa, and Takashi Suehiro. In-air knotting of rope by a dual-arm multi-finger robot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6202–6207. IEEE, 2015.

[203] Dana Kulić and Elizabeth A Croft. Safe planning for human-robot interaction. *Journal of Robotic Systems*, 22(7):383–396, 2005.

[204] Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

[205] Andrew M Ladd and Lydia E Kavraki. Using motion planning for knot untangling. *The International Journal of Robotics Research*, 23(7-8):797–808, 2004.

[206] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 2022.

[207] Kim G Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and computation*, 94(1):1–28, 1991.

[208] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[209] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33:19884–19895, 2020.

[210] Przemyslaw A Lasota, Terrence Fong, Julie A Shah, et al. A survey of methods for safe human-robot interaction. *Foundations and Trends® in Robotics*, 5(4):261–349, 2017.

[211] Przemyslaw A Lasota, Gregory F Rossano, and Julie A Shah. Toward safe close-proximity human-robot interaction with standard industrial robots. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 339–344. IEEE, 2014.

[212] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms.* Cambridge University Press, 2020.

[213] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. *Algorithmic and Computational Robotics*, pages 303–307, 2001.

[214] Neil Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, 16, 2003.

[215] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020.

[216] Alex X Lee, Max A Goldstein, Shane T Barratt, and Pieter Abbeel. A non-rigid point and normal registration algorithm with applications to learning from demonstrations. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 935–942. IEEE, 2015b.

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

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

[219] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

[220] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[221] Kuang-Huei Lee, Ian Fischer, Anthony Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive information accelerates learning in rl. *Advances in Neural Information Processing Systems*, 33:11890–11901, 2020.

[222] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[223] Michelle A Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950. IEEE, 2019.

[224] Robert Lee, Daniel Ward, Vibhavari Dasagi, Akansel Cosgun, Juxi Leitner, and Peter Corke. Learning arbitrary-goal fabric folding with one hour of real robot experience. In *Conference on Robot Learning*, pages 2317–2327. PMLR, 2021.

[225] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[226] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. *Advances in neural information processing systems*, 27, 2014.

[227] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. *Advances in neural information processing systems*, 24, 2011.

[228] Shen Li, Nadia Figueroa, Ankit J Shah, and Julie A Shah. Provably safe and efficient motion planning with uncertain human dynamics. In *Robotics: Science and Systems*, 2021.

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

[230] Terry J Ligocki and James A Sethian. Recognizing knots using simulated annealing. *Journal of Knot Theory and Its Ramifications*, 3(04):477–495, 1994.

[231] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

[232] Xingyu Lin, Harjatin Singh Baweja, and David Held. Reinforcement learning without ground-truth state. *arXiv preprint arXiv:1905.07866*, 2019.

[233] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, pages 256–266. PMLR, 2022.

[234] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[235] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *NIPS*, pages 206–214, 2012.

[236] Wen Hao Lui and Ashutosh Saxena. Tangled: Learning to untangle ropes with rgb-d perception. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 837–844. IEEE, 2013.

[237] Xiao Ma, David Hsu, and Wee Sun Lee. Learning latent graph dynamics for deformable object manipulation. *CoRR*, abs/2104.12149, 2021.

[238] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020.

[239] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

[240] Guilherme Maeda, Marco Ewerton, Takayuki Osa, Baptiste Busch, and Jan Peters. Active incremental learning of robot movement primitives. In *Conference on Robot Learning*, pages 37–46. PMLR, 2017.

[241] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. 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*, pages 2308–2315. IEEE, 2010.

[242] Daniel J. Mankowitz, Augustin Zídek, André Barreto, Dan Horgan, Matteo Hessel, John Quan, Junhyuk Oh, Hado van Hasselt, David Silver, and Tom Schaul. Unicorn: Continual learning with a universal, off-policy agent. *CoRR*, abs/1802.08294, 2018.

[243] Simon Manschitz, Jens Kober, Michael Gienger, and Jan Peters. Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. *Robotics and Autonomous Systems*, 74:97–107, 2015.

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

[245] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[246] Takamitsu Matsubara, Daisuke Shinohara, and Masatsugu Kidode. Reinforcement learning of a motor skill for wearing a t-shirt using topology coordinates. *Advanced Robotics*, 27(7):513–524, 2013.

[247] Takayuki Matsuno, Daichi Tamaki, Fumihito Arai, and Toshio Fukuda. Manipulation of deformable linear objects using knot invariants to classify the object condition based on image sensor information. *IEEE/ASME Transactions on Mechatronics*, 11(4):401–408, 2006.

[248] Roger McFarlane. A survey of exploration strategies in reinforcement learning. *McGill University*, 2018.

[249] Alberto Maria Metelli, Flavio Mazzolini, Lorenzo Bisi, Luca Sabbioni, and Marcello Restelli. Control frequency adaptation via action persistence in batch reinforcement learning. In *International Conference on Machine Learning*, pages 6862–6873. PMLR, 2020.

[250] Stephen Miller, Mario Fritz, Trevor Darrell, and Pieter Abbeel. Parametrized shape models for clothing. In *2011 IEEE International Conference on Robotics and Automation*, pages 4861–4868. IEEE, 2011.

[251] Stephen Miller, Jur Van Den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31(2):249–267, 2012.

[252] Hiroyuki Miyamoto, Stefan Schaal, Francesca Gandolfo, Hiroaki Gomi, Yasuharu Koike, Rieko Osu, Eri Nakano, Yasuhiro Wada, and Mitsuo Kawato. A kendama learning robot based on bi-directional theory. *Neural networks*, 9(8):1281–1302, 1996.

[253] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

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

[255] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[256] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. A framework for reinforcement learning and planning. *arXiv preprint arXiv:2006.15009*, 2020.

[257] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[258] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

[259] Takuma Morita, Jun Takamatsu, Koichi Ogawara, Hiroshi Kimura, and Katsushi Ikeuchi. Knot planning from observation. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 3, pages 3887–3892. IEEE, 2003.

[260] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[261] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation Letters*, 6(2):911–918, 2021.

[262] John M Murray. Single-ply pick-up devices. *AAMA Apparel Research J*, pages 87–98, 1975.

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

[264] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[265] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

[266] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[267] Tianwei Ni and Eric Jang. Continuous control on time. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022.

[268] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[269] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[270] Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[271] Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What matters for adversarial imitation learning? *Advances in Neural Information Processing Systems*, 34:14656–14668, 2021.

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

[273] Fumiaki Osawa, Hiroaki Seki, and Yoshitsugu Kamiya. Clothes folding task by tool-using robot. *Journal of Robotics and Mechatronics*, 18(5):618–625, 2006.

[274] Fumiaki Osawa, Hiroaki Seki, and Yoshitsugu Kamiya. Unfolding of massive laundry and classification types by dual manipulator. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 11(5):457–463, 2007.

[275] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[276] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International conference on machine learning*, pages 2701–2710. PMLR, 2017.

[277] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

[278] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos A. Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *Robotics: Science and Systems XIV*, 2018.

[279] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

[280] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR*, abs/1511.06342, 2016.

[281] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[282] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[283] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

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

[285] Emmanuel Pignat and Sylvain Calinon. Learning adaptive dressing assistance from human demonstration. *Robotics and Autonomous Systems*, 93:61–75, 2017.

[286] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[287] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted bellman residual minimization handling expert demonstrations. In *Joint European Conference on machine learning and knowledge discovery in databases*, pages 549–564. Springer, 2014.

[288] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

[289] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[290] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[291] Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pages 17811–17829. PMLR, 2022.

[292] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[293] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Visual adversarial imitation learning using variational models. *Advances in Neural Information Processing Systems*, 34:3016–3028, 2021.

[294] Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *CoRR*, abs/1610.01283, 2016.

[295] Ramya Ramakrishnan, Ece Kamar, Debadeepta Dey, Eric Horvitz, and Julie Shah. Blind spot detection for safe sim-to-real transfer. *Journal of Artificial Intelligence Research*, 67:191–234, 2020.

[296] Arnau Ramisa, Guillem Alenya, Francesc Moreno-Noguer, and Carme Torras. Using depth and appearance features for informed robot grasping of highly wrinkled clothes. In *2012 IEEE International Conference on Robotics and Automation*, pages 1703–1708. IEEE, 2012.

[297] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.

[298] Eric J Rawdon. Approximating the thickness of a knot. In *Ideal knots*, pages 143–150. World Scientific, 1998.

[299] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. {SQIL}: Imitation learning via reinforcement learning with sparse rewards. In *International Conference on Learning Representations*, 2020.

[300] Scott Reed, Aäron Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando Freitas. Parallel multiscale autoregressive density estimation. In *International conference on machine learning*, pages 2912–2921. PMLR, 2017.

[301] Kurt Reidemeister. *Knot theory*. BCS Associates, 1983.

[302] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[303] Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[304] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 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*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[305] Leonel Rozo, Joao Silverio, Sylvain Calinon, and Darwin G Caldwell. Learning controllers for reactive and proactive behaviors in human–robot collaboration. *Frontiers in Robotics and AI*, 3:30, 2016.

[306] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

[307] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[308] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[309] Fereshteh Sadeghi and Sergey Levine. (cad)\$ˆ2\$rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016.

[310] Mitul Saha and Pekka Isto. Motion planning for robotic manipulation of deformable linear objects. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2478–2484. IEEE, 2006.

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

[312] Khairul Salleh, Hiroaki Seki, Yoshitsugu Kamiya, and Masatoshi Hikizu. Inchworm robot grippers in clothes manipulation—optimizing the tracing algorithm. In *2007 International Conference on Intelligent and Advanced Systems*, pages 1051–1055. IEEE, 2007.

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

[314] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[315] Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[316] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In *Robotics research. the eleventh international symposium*, pages 561–572. Springer, 2005.

[317] Robert Glenn Scharein. *Interactive topological drawing.* Citeseer, 1998.

[318] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

[319] Riccardo Schiavi, Antonio Bicchi, and Fabrizio Flacco. Integration of active and passive compliance control for safe human-robot coexistence. In *2009 IEEE International Conference on Robotics and Automation*, pages 259–264. IEEE, 2009.

[320] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[321] Jonathan Scholz and Mike Stilman. Combining motion planning and optimization for flexible robot manipulation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 80–85. IEEE, 2010.

[322] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[323] John Schulman, Ankush Gupta, Sibi Venkatesan, Mallory Tayson-Frederick, and Pieter Abbeel. A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4111–4117. IEEE, 2013.

[324] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. Learning from demonstrations through the use of non-rigid registration. In *Robotics Research*, pages 339–354. Springer, 2016.

[325] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[326] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[327] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2021.

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

[329] Daniel Seita, Nawid Jamali, Michael Laskey, Ajay Kumar Tanwani, Ron Berenstein, Prakash Baskaran, Soshi Iba, John Canny, and Ken Goldberg. Deep transfer learning of pick points on fabric for robot bed-making. In *The International Symposium of Robotics Research*, pages 275–290. Springer, 2019.

[330] Daniel Seita, Justin Kerr, John Canny, and Ken Goldberg. Initial results on grasping and lifting physical deformable bags with a bimanual robot. In *IROS Workshop on Robotic Manipulation of Deformable Objects in Real-world Applications*, volume 2, page 3, 2021b.

[331] Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pages 19561–19579. PMLR, 2022.

[332] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[333] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *CoRR*, abs/1612.07307, 2016.

[334] Kyriacos Shiarlis, João V. Messias, and Shimon Whiteson. Inverse reinforcement learning from failure. In *Adaptive Agents and Multi-Agent Systems*, 2016.

[335] Daisuke Shinohara, Takamitsu Matsubara, and Masatsugu Kidode. Learning motor skills with non-rigid materials by reinforcement learning. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2676–2681. IEEE, 2011.

[336] Jonathan K Simon. Energy functions for polygonal knots. *Journal of Knot Theory and its Ramifications*, 3(03):299–320, 1994.

[337] Avi Singh, Larry Yang, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In *Robotics: Science and Systems*, 2019.

[338] Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020.

[339] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[340] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

[341] Jan Stria, Daniel Průša, and Vaclav Hlavac. Polygonal models for clothing. pages 173–184, 09 2014a.

[342] Jan Stria, Daniel Průša, Václav Hlaváč, Libor Wagner, Vladimír Petrík, Pavel Krsek, and Vladimír Smutný. Garment perception and its folding using a dual-arm robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 61–67, 2014b.

[343] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *ICLR (Poster)*, 2018.

[344] Hao Sun, Zhizhong Li, Xiaotong Liu, Bolei Zhou, and Dahua Lin. Policy continuation with hindsight inverse dynamics. *Advances in Neural Information Processing Systems*, 32, 2019.

[345] Li Sun, Gerarado Aragon-Camarasa, Paul Cockshott, Simon Rogers, and J Paul Siebert. A heuristic-based approach for flattening wrinkled clothes. In *Conference Towards Autonomous Robotic Systems*, pages 148–160. Springer, 2013.

[346] Li Sun, Gerardo Aragon-Camarasa, Simon Rogers, and J Paul Siebert. 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)*, pages 185–192. IEEE, 2015.

[347] Wen Sun, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning*, pages 3309–3318. PMLR, 2017.

[348] Priya Sundaresan, Ken Goldberg, and Joseph Gonzalez. Robotic untangling and disentangling of cables via learned manipulation and recovery strategies. 2021.

[349] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Jeffrey Ichnowski, Ellen R. Novoseller, Minho Hwang, Michael Laskey, Joseph E. Gonzalez, and Ken Goldberg. Untangling dense non-planar knots by learning manipulation features and recovery policies. *CoRR*, abs/2107.08942, 2021.

[350] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E Gonzalez, and Ken Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE, 2020.

[351] Jaeyong Sung, Ian Lenz, and Ashutosh Saxena. Deep multimodal embedding: Manipulating novel objects with point-clouds, language and trajectories. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2794–2801. IEEE, 2017.

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

[353] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[354] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[355] Kanata Suzuki, Momomi Kanamura, Yuki Suga, Hiroki Mori, and Tetsuya Ogata. In-air knotting of rope using dual-arm robot based on deep learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6724–6731. IEEE, 2021.

[356] Takahiro Suzuki, Yuji Ebihara, and Ken Shintani. Dynamic analysis of casting and winding with hyper-flexible manipulator. In *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 64–69. IEEE, 2005.

[357] Masaru Takizawa, Zhuonan Yao, Hiromu Onda, Shunsuke Kudoh, and Takashi Suehiro. Learning from observation of tabletop knotting using a simple task model. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 85–91. IEEE, 2019.

[358] Tomoya Tamei, Takamitsu Matsubara, Akshara Rai, and Tomohiro Shibata. Reinforcement learning of clothing assistance with a dual-arm robot. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 733–738. IEEE, 2011.

[359] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[360] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

[361] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[362] Yadong Teng, Huimin Lu, Yujie Li, Tohru Kamiya, Yoshihisa Nakatoh, Seiichi Serikawa, and Pengxiang Gao. Multidimensional deformable object manipulation based on dn-transporter networks. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[363] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

[364] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[365] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

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

[367] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Natalia Díaz-Rodríguez, and David Filliat. Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer. *arXiv preprint arXiv:1906.04452*, 2019.

[368] Lukas Twardon and Helge Ritter. Active boundary component models for robotic dressing assistance. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2811–2818. IEEE, 2016.

[369] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Pieter Abbeel, Sergey Levine, Kate Saenko, and Trevor Darrell. Adapting deep visuomotor representations with weak pairwise constraints. In *Algorithmic Foundations of Robotics XII*, pages 688–703. Springer, 2020.

[370] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[371] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[372] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[373] Trinh Van Vinh, Tetsuo Tomizawa, Shunsuke Kudoh, and Takashi Suehiro. A new strategy for making a knot with a general-purpose arm. In *2012 IEEE International Conference on Robotics and Automation*, pages 2217–2222. IEEE, 2012.

[374] Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017.

[375] Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[376] Vainavi Viswanath, Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Ellen Novoseller, Jeffrey Ichnowski, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. Disentangling dense multi-cable knots. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3731–3738. IEEE, 2021.

[377] Nelson Vithayathil Varghese and Qusay H Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9):1363, 2020.

[378] Hidefumi Wakamatsu, Eiji Arai, and Shinichi Hirai. Knotting/unknotting manipulation of deformable linear objects. *The International Journal of Robotics Research*, 25(4):371–395, 2006a.

[379] Hidefumi Wakamatsu, Akira Tsumaya, Eiji Arai, and Shinichi Hirai. Planning of one-handed knotting/raveling manipulation of linear objects. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 2, pages 1719–1725. IEEE, 2004.

[380] Hidefumi Wakamatsu, Akira Tsumaya, Eiji Arai, and Shinichi Hirai. Manipulation planning for unraveling linear objects. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2485–2490. IEEE, 2006b.

[381] Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. Learning to reinforcement learn. *CoRR*, abs/1611.05763, 2016.

[382] Weifu Wang and Devin Balkcom. Tying knot precisely. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3639–3646. IEEE, 2016.

[383] Weifu Wang, Dmitry Berenson, and Devin Balkcom. An online method for tight-tolerance insertion tasks for string and rope. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2488–2495. IEEE, 2015.

[384] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[385] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[386] Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, pages 192–202. PMLR, 2022.

[387] Zehang Weng, Fabian Paus, Anastasiia Varava, Hang Yin, Tamim Asfour, and Danica Kragic. Graph-based task-specific prediction models for interactions between deformable and rigid objects. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5741–5748. IEEE, 2021.

[388] Bryan Willimon, Stan Birchfield, and Ian Walker. Classification of clothing using interactive perception. In *2011 IEEE International Conference on Robotics and Automation*, pages 1862–1868. IEEE, 2011.

[389] Bryan Willimon, Stan Birchfield, and Ian Walker. Model for unfolding laundry using interactive perception. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4871–4876. IEEE, 2011.

[390] Adrian Wong, Andy Zeng, Arnab Bose, Ayzaan Wahid, Dmitry Kalashnikov, Ivan Krasin, Jake Varley, Johnny Lee, Jonathan Tompson, Maria Attarian, Pete Florence, Robert Baruch, Sichun Xu, Stefan Welker, Vikas Sindhwani, Vincent Vanhoucke, and Wayne Gramlich. Pyreach - python client sdk for robot remote control. `https://github.com/google-research/pyreach`, 2022.

[391] Bohan Wu, Suraj Nair, Roberto Martin-Martin, Li Fei-Fei, and Chelsea Finn. Greedy hierarchical variational autoencoders for large-scale video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2318–2328, 2021.

[392] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019.

[393] Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. In *International Conference on Learning Representations*, 2020.

[394] Danfei Xu and Misha Denil. Positive-unlabeled reward learning. In *Conference on Robot Learning*, pages 205–219. PMLR, 2021.

[395] Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Dextairity: Deformable manipulation can be a breeze. In *Proceedings of Robotics: Science and Systems (RSS)*, 2022.

[396] Yuji Yamakawa, Akio Namiki, and Masatoshi Ishikawa. Motion planning for dynamic knotting of a flexible rope with a high-speed robot arm. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 49–54. IEEE, 2010.

[397] Yuji Yamakawa, Akio Namiki, Masatoshi Ishikawa, and Makoto Shimojo. 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*, pages 2691–2696. IEEE, 2008.

[398] Kimitoshi Yamazaki and Masayuki Inaba. A cloth detection method based on image wrinkle feature for daily assistive robots. In *MVA*, pages 366–369, 2009.

[399] Kimitoshi Yamazaki, Ryosuke Oya, Kotaro Nagahama, and Masayuki Inaba. A method of state recognition of dressing clothes based on dynamic state matching. In *Proceedings of the 2013 IEEE/SICE International Symposium on System Integration*, pages 406–411. IEEE, 2013.

[400] Kimitoshi Yamazaki, Ryosuke Oya, Kotaro Nagahama, Kei Okada, and Masayuki Inaba. Bottom dressing by a life-sized humanoid robot provided failure detection and recovery functions. In *2014 IEEE/SICE international symposium on system integration*, pages 564–570. IEEE, 2014.

[401] Kimitoshi Yamazaki, Ryosuke Oya, Kotaro Nagahama, Kei Okada, and Masayuki Inaba. Bottom dressing by a dual-arm robot using a clothing state estimation based on dynamic shape changes. *International Journal of Advanced Robotic Systems*, 13(1):5, 2016.

[402] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. In *Conference on Robot Learning*, pages 564–574. PMLR, 2021.

[403] Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Jianye Hao, Zhaopeng Meng, and Peng Liu. Exploration in deep reinforcement learning: a comprehensive survey. *arXiv preprint arXiv:2109.06668*, 2021.

[404] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

[405] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[406] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *CoRR*, abs/1910.01741, 2019.

[407] Shun-Zheng Yu. Hidden semi-markov models. *Artificial intelligence*, 174(2):215–243, 2010.

[408] Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010. Special Review Issue.

[409] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[410] Wenhao Yu, Ariel Kapusta, Jie Tan, Charles C Kemp, Greg Turk, and C Karen Liu. Haptic simulation for robot-assisted dressing. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 6044–6051. IEEE, 2017.

[411] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

[412] Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

[413] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

[414] Fan Zhang, Antoine Cully, and Yiannis Demiris. Personalized robot-assisted dressing using user modeling in latent spaces. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3603–3610. IEEE, 2017.

[415] Fan Zhang, Antoine Cully, and Yiannis Demiris. Probabilistic real-time user posture tracking for personalized robot-assisted dressing. *IEEE Transactions on Robotics*, 35(4):873–888, 2019.

[416] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[417] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, pages 7444–7453. PMLR, 2019.

[418] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

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

[420] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[421] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

[422] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.