Leveraging Vision, Force Sensing, and Language Feedback for Deformable Object Manipulation

Zhanyi Sun

CMU-RI-TR-24-NN

June, 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To the people who inspired me.

Abstract

Deformable object manipulation represents a significant challenge in robotics due to its complex dynamics, lack of low-dimensional state representations, and severe self-occlusions. This challenge is particularly critical in assistive tasks, where safe and effective manipulation of various deformable materials can significantly improve the quality of life for individuals with disabilities and address the growing needs of an aging society. This thesis studies both specific applications in robot-assisted dressing and a generic framework for reinforcement learning-based manipulation of deformable objects.

In the first part, we present a robot-assisted dressing system capable of handling diverse garments and human body shapes and arm poses using reinforcement learning. By employing partial point cloud observations, policy distillation, and guided domain randomization, this work demonstrates effective policy learning that generalizes across various real-world assistive dressing scenarios. To enhance the safety and comfort of the dressing system, we further propose a novel multi-modal learning framework with vision and force sensing. We combine a vision-based reinforcement learning policy trained in simulation with a force dynamics model trained with real robot data to infer actions that facilitate the dressing process without applying excessive force on the person.

In the second part, we propose a novel framework that automatically generates reward functions for agents to learn new tasks by leveraging feedback from vision language foundation models. Since our method only requires a text description of the task goal and the agent's visual observations, bypassing the need for low-dimensional state representations of objects, it is particularly useful for deformable object manipulation tasks.

Acknowledgments

My first and deepest thanks go to my advisors, Prof. David Held and Prof. Zackory Erickson, for their invaluable guidance and support throughout my Master's journey. Their detailed feedback, constant encouragement, and unwavering dedication have been essential in overcoming challenges and refining the work presented in this thesis. From them, I have learned critical thinking, effective experimental design, and project management skills. Dave, your passion, curiosity, and persistence in research have been truly inspiring. As an advisor, you have a remarkable eye for detail while never losing sight of the larger picture. The philosophy and principles I learned from you will continue to guide my future research endeavors. Zackory, your enthusiasm for tackling big, challenging research problems in robotics and your innovative approach have been truly motivating. You have taught me to identify the right problems in robotics and address them with creative solutions. Your energy and ability to make every discussion engaging are qualities I always appreciate.

I also extend my heartfelt thanks to my Ph.D. student mentor, Yufei Wang, for your immense guidance and mentorship throughout my Master's journey. I am extremely fortunate and grateful for the opportunity to collaborate with you on numerous fun projects. From you, I have learned the delicate art of developing insightful research ideas and pursuing them with scientific rigor. Whenever I faced obstacles, your detailed and timely advice always provided the inspiration I needed. Your passionate and positive attitude toward research, along with your unwavering perseverance in the face of challenges, are qualities I hope to embody in my future endeavors.

I would like to thank my thesis committee members, Prof. Andrea Bajcsy and Xianyi Cheng, for taking the time out of your busy schedules to serve on my committee. I am truly grateful for your valuable feedback and suggestions on high-level research directions and the presentation of this thesis. Andrea, it has been a great pleasure to discuss research with you and benefit from your generous insights. Your advice on presenting research has been incredibly helpful to junior students like me. Your energy and warmth in every discussion are qualities I will always cherish. Xianyi, thank you for our many insightful discussions on control, planning, and learning in robotics. Your expertise in robotic manipulation and your passion for robotics research are qualities I greatly admire. I can't wait to see what you will achieve at Duke University as a professor!

Next, I would like to thank all the members of both the R-PAD and RCHI labs. It has been an honor and pleasure to work alongside such wonderful and talented labmates. I have learned a great deal from each of you and will miss our interactions.

I would like to give special thanks to my undergraduate advisors, Prof. Vaibhav Unhelkar, Prof. Lydia E. Kavraki, and Prof. Yingyan (Celine) Lin. They kindly introduced me to the fields of artificial intelligence and robotics, and I am always grateful for their invaluable guidance and encouragement.

Finally, I wish to thank all my friends in Pittsburgh for their constant support inside and outside of research. I am also deeply grateful to my family for their unconditional love and support in all my decisions. Their unwavering love, support, and encouragement have been a source of strength that fueled me to push through difficult times. I would not be here without them.

Funding

The work in this thesis was supported by the National Science Foundation under Grant No. IIS-2046491, and NVIDIA Corporation for their academic hardware grant. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or NVIDIA Corporation

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Learning successful policies for deformable object manipulation tasks is difficult because they challenge common assumptions in robotic manipulation, such as access to low-dimensional state representations, known dynamics, and minimal occlusion. Despite these challenges, deformable object manipulation is crucial in assistive robotics, where tasks like assistive feeding and dressing can significantly enhance the quality of life for individuals and meet the demands of an aging population. This thesis first takes robot-assisted dressing as a compelling case study due to its potential to improve the lives of the elderly and disabled, as well as alleviate the burden on caregivers. Then, we present a generic framework for reward and policy learning for deformable object manipulation tasks.

The task of robot-assisted dressing involves handling the variability of human body shapes, poses, and garment types, requiring a system that can generalize across diverse scenarios. This necessitates the development of algorithms and systems capable of robustly handling these variations. Additionally, ensuring the safety and comfort of the individual during the dressing process is critical, as improper handling can cause discomfort or injury.

To address these challenges, we develop a comprehensive robot-assisted dressing system with a focus on two primary objectives: generalization and safety (Chapter 2). To enable generalization, our approach utilizes partial point cloud observations and policy distillation, along with guided domain randomization for sim-to-real transfer. These techniques enable the system to learn effective dressing policies that can generalize across a wide range of real-world scenarios. The safety challenge is addressed through a multi-modal learning framework that combines vision and force sensing. By integrating a vision-based reinforcement learning (RL) policy trained in simulation with a force dynamics model learned from real-world data, our system can perform dressing actions that minimize applied force, thereby enhancing safety and comfort for the person being dressed.

However, learning successful policies for the specific task of robot-assisted dressing requires extensive human engineering and iterative tuning. Designing reward functions for tasks with high-dimensional environments and observations, such as cloth manipulation, can be very challenging. To alleviate this burden, we introduce a novel framework for automatic reward generation (Chapter 3). This framework leverages feedback from vision language foundation models (VLMs) to automatically generate reward functions based on a text description of the task goal and visual observations. Our method significantly reduces the need for low-dimensional state representations and extensive manual reward tuning, making it particularly advantageous for deformable object manipulation tasks.

1.1 Thesis Organization

This thesis is organized as follows:

- In Chapter 2, we introduce a robot-assisted dressing system, which comprises two projects. In the first project, named "One Policy to Dress Them All," we aim to tackle the generalization challenge in robot-assisted dressing. Many prior works attempt to solve this task, but they make certain assumptions such as a fixed garment and a fixed arm pose that limit their ability to generalize [17, 34, 35, 48, 50, 64, 129, 130]. In this project, we develop a system that is able to dress different garments on people with diverse body shapes and poses. We show that with proper design of the policy architecture, RL can be used to learn effective dressing policies with partial point cloud observations. We further leverage policy distillation to combine multiple policies trained on different ranges of human arm poses into a single policy that works over a wide range of different arm poses. Our system is able to dress 86% of the length of the participants' arms on average, validated by 510 dressing trials in a human study with 17 participants with different arm poses and dressed garments.

To further address the safety challenge in the second project, we propose FCVP (**F**orce-**C**onstrained **V**isual **P**olicy), a novel technique that leverages both vision and force modalities for assistive dressing. Our approach first trains a vision-based dressing policy using RL in simulation with varying body sizes, poses, and types of garments for the purpose of generalization. We then learn a force dynamics model for action planning to ensure safety. Prior works either study force sensing for robot-assisted dressing in simulation only [27, 106], or use system identification to tune simulation parameters to approximate real robot force sensing [123]. Our work differs from these by learning a force dynamics model directly from real-world data to combat the limitations of simulating accurate force data when deformable garments interact with the human body. Our proposed method combines the vision-based policy, trained in simulation, with the force dynamics model, learned in the real world, by solving a constrained optimization problem to infer actions that facilitate the dressing process without applying excessive force on the person. When evaluated in a real-world human study with 10 participants across 240 dressing trials, our method outperforms baselines in both force violations and dressing performance.

- In Chapter 3, we introduce RL-VLM-F (**R**einforcement **L**earning from **V**ision **L**anguage Foundation **M**odel **F**eedback), a method that automatically generates reward functions for agents to learn new tasks, using only a text description of the task goal and the agent's visual observations, by leveraging feedback from vision language foundation models (VLMs). The key to our approach is to query VLMs to give preferences over pairs of the agent's image observations based on the text description of the task goal, and then learn a reward function from the preference labels, rather than directly prompting these models to output a raw reward score as done in prior work, which can be noisy and inconsistent [70, 76, 92, 99]. We demonstrate that RL-VLM-F successfully produces effective rewards and policies across various domains — including classic control, as well as manipulation of rigid, articulated, and deformable objects — without the need for human supervision, outperforming prior methods that use large pretrained models for reward generation under the same assumptions.

Chapter 2

Robot Learning for Assistive Dressing

2.1 One Policy to Dress Them All

2.1.1 Introduction

Robot-assisted dressing could benefit the lives of many people. Dressing is a core activity of daily living.

A 2016 study by the National Center for Health Statistics [41] shows that 92% of all residents in nursing facilities and at-home care patients require assistance with dressing. Such needs will likely keep growing due to aging populations. A robot-assisted dressing system could not only improve life quality for older adults and individuals with disabilities by helping them maintain independence and privacy, but also help mitigate the growing shortage of nursing staff, and provide some much needed respite for caregivers.

Despite its potential societal impact, robot-assisted dressing remains a challenging task for robotics for the following reasons. As in many deformable object manipulation tasks, there is no compact state representation of cloth, and the dynamics of garments are non-linear and complex [131]. Compared to table-top cloth manipulation tasks such as folding and smoothing that can be solved via pick-and-place actions, the dressing task demands more dexterous manipulation actions in 3D space.

Furthermore, people with disabilities and older adults usually have a limited range of motion and thus the dressing robot needs to generalize to a diverse range of arm poses. Finally, there are many different types of garments with varying geometries and material properties, e.g., short-sleeve hospital gowns and long-sleeve jackets. It is non-trivial for a robot to be able to assist in dressing all these garments, since dressing a slender long-sleeve elastic jacket might require very different end-effector trajectories compared with dressing a wide, short-sleeve non-elastic hospital gown.

All these factors render robot-assisted dressing a very challenging problem.

There have been many prior works that investigate robot-assisted dressing, yet they make certain assumptions that limit their ability to generalize. Most prior works [28, 29, 34, 35, 47, 50, 130] assume that the robot is dressing a known fixed garment and thus does not generalize to dressing diverse garments. Other prior works simplify the assistive dressing problem by assuming that the person holds a single pose [28, 89], or can move their arm into a pose that has high probability of dressing success [35, 47, 50]. We aim to improve upon these prior works to build a dressing system that can generalize to diverse garments and human poses.

In this work, we learn a single policy that is able to dress diverse garments on different people holding diverse poses from partial point cloud observations using a single depth camera. We use point clouds as input to our policy in order to represent and generalize to different garments and human arm poses. We show that with proper design of the policy and Q-function architectures,

reinforcement learning (RL) can be used to learn effective policies with partial point cloud observations that work well for dressing diverse garments. We further leverage policy distillation to combine multiple policies trained on different ranges of human arm poses into a single one that works over a wide variety of different poses. Finally, for robust sim2real transfer, we employ "guided domain randomization", which trains a policy with randomized observations by imitating policies trained without any randomization. The final domain-randomized policy can be reliably transferred to a real robot manipulator and dress real people. We conducted a human study with 17 participants, and performed 510 total dressing trials with different arm poses and garments. On average, our system is able to dress 86% of the length of the participants' arms, and achieves a statistically significant difference in responses as compared to alternative baselines in a 7-point Likert item.

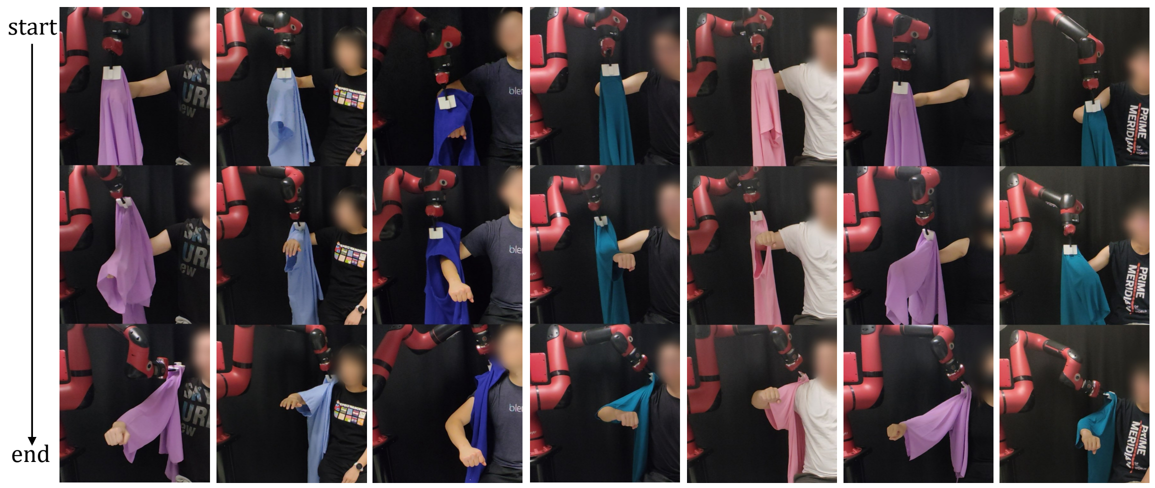


Figure 2.1: We develop a robot-assisted dressing system based on a single learned policy that is able to dress different people with diverse poses and garments. Each column shows snapshots from a trajectory of our policy. We learn the dressing policy using reinforcement learning with point cloud observations to generalize to diverse garments. We use policy distillation to combine multiple policies that each work for a small range of arm poses into a single policy that works for a large variety of arm poses.

We also provide a comprehensive analysis of our dressing system's performance in the human study to understand its strength and weakness.

In summary, we make the following contributions:

- We develop a robot-assisted dressing system based on a learned policy with partial point cloud observations that generalizes to different people, arm poses, and garments.
- We show the effectiveness of policy distillation to increase the effective working range of the policy.
- We perform comprehensive real-world evaluations of our system on a manikin, as well as with 510 dressing trials in a human study with 17 participants of varying arm poses and dressed garments.

2.1.2 Related Work

Robot Assisted Dressing

There have been many prior works in robot-assisted dressing [128] which can be categorized as follows. Some works make the assumption that the person being dressed is collaborative in performing the dressing task [22, 50], while others do not [28, 35]. We also do not assume a collaborative human in this work. Instead, we simplify the problem by assuming that the human is holding a fixed pose throughout the dressing procedure.

A large body of works focus on user modeling and building a personalized dressing plan for each human participant [17, 34, 35, 48, 50, 64, 129, 130]. In contrast to these prior works, we do not focus on user modeling and aim to learn a single policy that is able to generalize to diverse poses and body sizes of different human participants. We leave the integration of user modeling for future work.

Another line of work studies haptic perception and simulation during robot-assisted dressing [27, 49, 106, 123]. In contrast, we demonstrate how a robot can perform dressing assistance using only partial point clouds from a single off-the-shelf depth camera. Some prior works focus on learning where along a garment to grasp in preparation for dressing [89, 127]. In this work we make the assumption that the garment has already been grasped by the robot, but our method can be combined with these prior work to remove this assumption.

Reinforcement learning has been used in some prior works for dressing backpacks [47] or hospital gowns [22]. However, they assume a fixed garment and thus the policy has limited generalization towards different garments. Our policy can generalize to dressing diverse garments as we use point cloud as the garment representation.

Robotic Deformable Object Manipulation

Deformable object manipulation has long been a core task for robotics. It is challenging due to the complicated dynamics of deformables, high-dimensional state representation, and perception complexities such as self-occlusion. Many prior deformable object manipulation tasks such as cloth smoothing [38, 67, 115, 119], cloth folding [4, 59, 110], bedding manipulation [86, 95], dough rolling [66], rope reconfiguration [65, 96], and bag manipulation [5, 19, 96] focus on the 2D table-top setting. Many of these prior cloth manipulation tasks can be solved by simple pick-and-place actions or the use of other pre-defined motion primitives [38, 44, 67, 95, 96, 110, 115, 119]. Our work looks at the challenging problem of dressing assistance, which involves manipulating a deformable garment suspended in 3D space with complex closed-loop actions during physical human-robot interaction.

Policy Learning for Manipulation From Point Clouds

There has been much recent work that aims to learn robotic manipulation policies from point cloud observations. Some of these works propose new algorithms for imitation learning from point clouds for grasping and manipulating tools or articulated objects [26, 82, 97, 116]. Some recent works explore applying RL with point cloud observations for grasping or dexterous hand manipulation with rigid objects [45, 68, 90, 105]. Our work differs from these as we apply RL from point cloud observations for deformable cloth manipulation.

2.1.3 Task Definitions and Assumptions

As shown in Fig. 2.13, the task we study in this paper is single-arm dressing, where the goal is to fully dress the sleeve of a garment onto a person's arm, and the task is considered to be complete when the sleeve of the garment covers the person's shoulder. The person can hold different arm poses before the dressing starts. The arm pose is defined by three joint angles $\phi = [\phi_1, \phi_2, \phi_3] \in \mathbb{R}^3$, where ϕ_1 is the lifting and lowering angle of the shoulder, ϕ_2 is the inwards-outwards bending angle of the elbow towards the body, and ϕ_3 is the lifting and lowering angle of the elbow (Fig. 2.5 illustrates these joint angles). A depth camera is used to record depth images of the scene, and partial point clouds P can be computed from the depth image. The ultimate goal of the paper is to learn a policy π to dress diverse garments $g \in G$ for people holding a diverse range of arm poses $\phi \in \Phi$. At each time step, the policy takes as input the point cloud P and outputs an action $a = \pi(P)$ as the delta transformation for the robot end-effector. The garment set G we consider includes hospital gowns and common everyday garments such as vests and cardigans with different sleeve lengths, geometries, and materials. The set of arm poses Φ is specified by the min and max

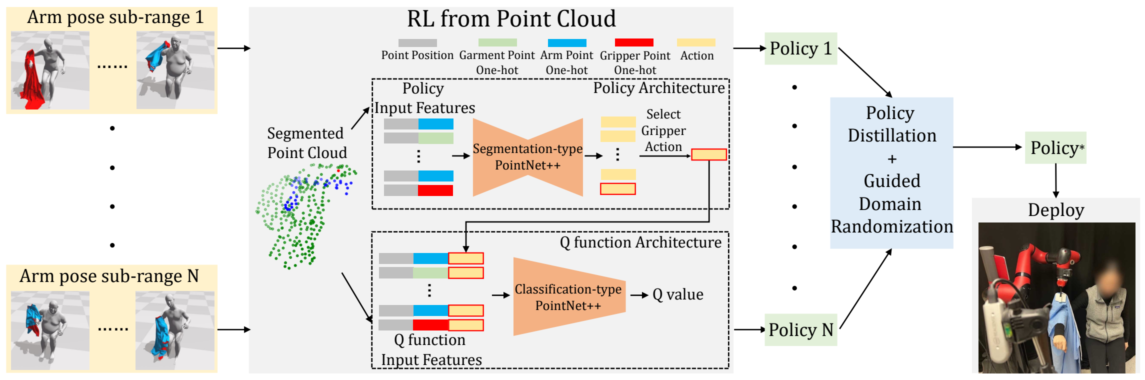


Figure 2.2: An overview of our proposed robot-assisted dressing system. (Left) In simulation, we divide the diverse arm pose range into multiple sub-ranges to ease policy learning. (Middle) We propose a new policy architecture and a corresponding Q function architecture for reinforcement learning from partial point cloud observations to learn effective dressing policies on each of the divided pose sub-ranges. (Right) We then leverage policy distillation to combine policies working on different pose sub-ranges into a single policy that works for a diverse range of poses. We also perform guided domain randomization for sim2real transfer, and we deploy the distilled policy to a real-world human study dressing real people.

values for each joint angle: $\Phi = \{[\phi_1, \phi_2, \phi_3] \mid \phi_i \in [\phi_i^{min}, \phi_i^{max}], i = 1, 2, 3\}$. Fig. 2.5 shows the garments and arm poses in Φ that we test in the real-world human study.

We make two assumptions for the dressing task. First, we assume that the robot has already grasped a part of the garment around the opening of the garment shoulder in preparation for dressing, since grasping is not the focus of our work. Besides, there has been some prior work that learn where along a garment to grasp for dressing [89, 127], and our system can be combined with these prior work to remove this assumption. This assumption has been used in prior work as well [28, 50]. Second, we assume that the person holds the pose static during the dressing process. This assumption helps address the visual occlusion of the arm caused by the cloth during the dressing process; with this assumption we can obtain the static arm point cloud before the dressing starts. This assumption has also been validated in other studies with participants with impairments [50]. We leave for future work adapting to human motion during the dressing process.

2.1.4 Method

System Overview

As illustrated in Fig. 2.2, our robot-assisted dressing system consists of the following components. We first use reinforcement learning (RL) to learn the dressing policy π in a physics-based simulation by formulating the robot assisted dressing problem as a Partially Observable Markov Decision Process (POMDP). We design a special policy network architecture for efficient training of effective RL policies from partial point cloud observations. As it is hard to train a single policy that generalizes to a diverse range of arm poses Φ, we divide the arm pose range into multiple sub-ranges $\{\Phi^{sub}\}_{i=1}^{N}$ and train a policy π_i on each of them. We then use policy distillation to combine these different policies into a single policy π^* that works for the wide range of arm poses Φ. For robust sim2real transfer, we further train the policy π^* with domain randomization in the policy observation, with a behaviour cloning loss to imitate policies that are trained without any randomization, a procedure which we name as "guided domain randomization".

Finally, we deploy the domain-randomized policy to a real robot that successfully dresses different participants with diverse poses and garments in a human study.

Learning to Dress with Reinforcement Learning

We use reinforcement learning in simulation to train policies for the dressing task. We formulate the robot-assisted dressing task as a POMDP $\langle S, A, O, R, T, U \rangle$, where S is the state space, A is the action space, O is the observation space, R is the reward function, T is the transition dynamics, and U is the measurement function that generates the observation from the state. We now detail the design of the core components of the POMDP as follows.

Observation Space O: Due to the lack of a compact state representation for cloth, the garment naturally requires a high-dimensional representation such as an image, mesh, or point cloud. To facilitate sim2real transfer, we use the partial point cloud P of the dressing scene computed from a depth camera as the policy observation.

We perform cropping on the point cloud to only keep points that belong to the right arm of the human, denoted as P^h, and those that belong to the garment, denoted

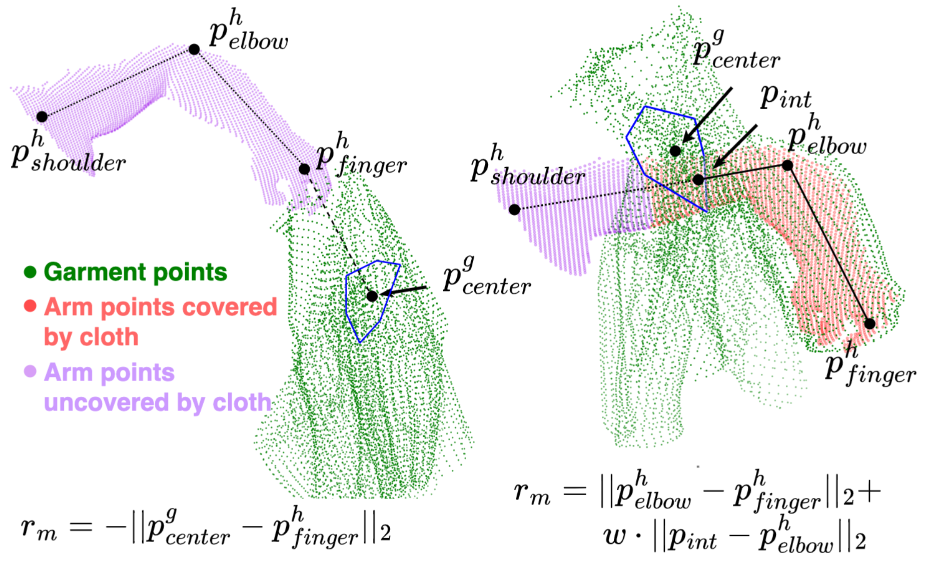

p^h_{elbow}
$p^h_{shoulder}$
p^h_{finger}
p^g_{center}
● Garment points
● Arm points covered by cloth
● Arm points uncovered by cloth
$r_m = -||p^g_{center} - p^h_{finger}||_2$
p^g_{center}
p_{int}
p^h_{elbow}
$p^h_{shoulder}$
p^h_{finger}
$r_m = ||p^h_{elbow} - p^h_{finger}||_2 + w \cdot ||p_{int} - p^h_{elbow}||_2$

Figure 2.3: Illustration of the main reward for the dressing task. (Left) The garment is not dressed onto the person's arm. (Right) The garment is dressed onto the person's arm.

as P^g. In simulation we can easily perform such cropping using the ground-truth simulator information; see Section 2.2.6 for details on how we perform such cropping in the real world. We further add a single point at the position of the robot end-effector to the point cloud to represent the robot gripper, denoted as P^r. The policy observation is then the concatenation of the arm, garment, and robot gripper points: $o = [P^h; P^g; P^r]$. See the middle part of Fig. 2.2 for an example of the segmented point cloud.

Action Space A**:** The action $a \in SE(3)$ is the delta transformation of the robot end-effector. We represent the delta transformation as a 6D vector, where the first 3 components are the delta translation, and the second 3 components describe the delta rotation using axis-angle. We set the roll rotation to be 0 in the action as it is not necessary for the dressing task.

Reward Function R: Fig. 2.3 illustrates the main reward r_m we use for the dressing task, which measures the progress of the task. The dressing task can be divided to two phases.

Before the garment is dressed onto the person's forearm, the reward r_m is the negative distance from the garment shoulder opening center p^g_{center} to the finger of the person p^h_{finger}: $r_m = -||p^g_{center} - p^h_{finger}||_2$. After the garment is dressed onto the person's forearm, we compute the reward r_m based on the distance the garment has been dressed onto the person's arm. To compute the dressed distance, we first approximate the opening of the garment shoulder as a hexagon (shown in blue in Fig. 2.3). Then, we represent the person's forearm as a line that connects the elbow point p^h_{elbow} and the finger point p^h_{finger}, and we represent the upper arm as another line that connects the shoulder point $p^h_{shoulder}$ and the elbow point p^h_{elbow}. Next, the intersection point p_{int} between the garment shoulder opening hexagon and the two arm lines are computed. If the intersection point p_{int} is on the forearm, r_m equals the dressed distance, which is the distance between the intersection point and the person's finger point: $r_m = ||p_{int} - p^h_{finger}||_2$. If the intersection point p_{int} is on the upper arm, the dressed distance is the length of the forearm plus the distance between the intersection point and the elbow point, and r_m is a weighted combination of these two: $r_m = ||p^h_{elbow} - p^h_{finger}||_2 + w \cdot ||p_{int} - p^h_{elbow}||_2$. We set $w = 5$ to encourage the policy to turn at the elbow and dress the upper arm.

In addition to the main reward term that measures the task progression, we also have three additional reward terms. The first is a force penalty r_f that prevents the robot from applying too much force through the garment to the person. The second is a contact penalty r_c that prevents the robot end-effector from moving too close to the person. The last reward term is a deviation penalty r_d that discourages the garment center from moving too far away from the arm. The full reward is given by: $r = r_m + r_f + r_c + r_d$. More details of how these terms are computed can be found in the Appendix A.1.1.

Note that the reward function is only available in simulation, as it requires access to the ground-truth garment and human mesh information, which is non-trivial to estimate in the real world.

Policy and Q function Architecture: Most prior works [68, 90, 105] that train RL policies with point cloud observation use a classification-type PointNet-like [87, 88] network architecture for the policy, which encodes the whole partial point cloud to a single action vector. There have been some recent works showing that instead of compressing the whole point cloud into a single action vector, inferring the action from a dense output leads to better performance [18, 26, 38, 97, 113, 114, 125, 126].

We follow the dense action representation idea and propose a new policy architecture named *Dense Transformation* for reinforcement learning from point clouds.

As shown in Fig. 2.2, the input to the policy is a segmented point cloud $P = \{p_i\}_{i=1}^{M}$ of size M, which contains the garment points P^g, human right arm points P^h, and a single point at the robot end-effector position representing the robot gripper P^r. The features for each point p_i include its 3D position, and a 3-dimensional one-hot vector indicating the class of the point, i.e., whether the point belongs to the garment, the human arm, or the robot gripper. Instead of using a classification-type neural network architecture (e.g., a classification-type PointNet++) that compresses the whole point cloud into a single action vector, the Dense Transformation policy uses a segmentation-type neural network architecture (e.g., a segmentation-type PointNet++) that outputs a dense per-point action vector $\{a_i\}_{i=1}^{M}$ (see Fig. 2.2). Among these M action vectors, we only execute the action vector a^* corresponding to the gripper point P^r, i.e, we select the action $a^* = a_j$, where j is the index of the gripper point. We could alternatively use a classification-type PointNet++ that encodes the whole point cloud to a single action, but we find that using a segmentation-type network leads to slightly better performance (though the difference is relatively small).

For the Q function, given the current point cloud P and the action sampled from the Dense Transformation policy $a^* \sim \pi(P)$, we concatenate a^* as an additional feature to every point p_i in the input point cloud, so the feature of each point includes its 3D position, the one-hot vector of the class type, and the action a^*. We then use a classification-type neural network architecture (i.e., classification-type PointNet++) to output a scalar Q value. This Q function architecture has also been used in prior work [105], although with a different policy architecture. We compare to other ways of representing the Q function in experiments and show that this one works the best. We use SAC [39] as the underlying RL algorithm, and we use PointNet++ [88] for the policy and Q function network.

Generalization to Diverse Human Poses via Policy Distillation

Learning a dressing policy that works on a diverse range of arm poses can be viewed as a multi-task learning problem, where each small range of poses can be considered as an individual task. The same holds true for learning a policy that works for diverse garments – each garment can be treated as an individual task. In our experiments, we find it is possible to learn a single universal policy for a diverse set of garments, and the performance is similar to that of learning an individual policy for each garment. However, we find it challenging to learn a single policy that works well for a diverse range of arm poses, possibly due to imbalanced learning speed for different tasks (e.g., some poses are easier to learn compared with others), conflicting gradients from different tasks (the desired trajectory for one arm pose might contradict to another), and other potential issues. Inspired by [93, 94], we use policy distillation to address this issue.

Policy Distillation. Given a set of policies that were each trained for a single task, policy distillation [93] can be used to combine the set of policies into a single policy that works for all of the tasks. It has been shown that this often outperforms directly training a single policy for all of the tasks. In our case, we train individual policies each for dressing a human arm pose sub-range; we then distill these policies into a single policy that works for the full diverse range of arm poses Φ.

To employ policy distillation, we first need to decompose the diverse arm pose range Φ into smaller ranges $\{\Phi_i^{sub}\}_{i=1}^N$ where RL can be directly used to learn effective policies on these smaller pose ranges. As aforementioned, the arm pose range Φ is specified by min and max values for each of the three joint angles $\Phi = \{[\phi_1, \phi_2, \phi_3] \mid \phi_i \in [\phi_i^{min}, \phi_i^{max}], i = 1, 2, 3\}$. We perform the decomposition by dividing each joint angle range $[\phi_i^{min}, \phi_i^{max}]$ into a number of smaller intervals. For example, we can uniformly divide the range $[\phi_i^{min}, \phi_i^{max}]$ to $\Phi_i^j = [\phi_i^{min} + (j-1)\delta, \phi_i^{min} + j\delta], j = 1, ..., L$, where $\delta = (\phi_i^{max} - \phi_i^{min})/L$. This will then result in L^3 sub-ranges $\{\Phi^{sub} = \Phi_1^j \times \Phi_2^k \times \Phi_3^l\}_{j=1,k=1,l=1}^L$.

After we decompose the diverse arm pose range Φ into N smaller ranges, we train N "teacher" policies $\{\pi_i^t\}_{i=1}^N$, one for each pose sub-range, using RL as described in Section 2.1.4. We then distill these N teacher policies $\{\pi_i^t\}_{i=1}^N$ into a single "student" policy π_s by training the student policy with a combination of the RL loss and a policy distillation loss, shown in Eq. 2.1.

Specifically, let θ_s denote the parameters of the student policy π_s, let θ_t^i denote the parameters of the i^{th} teacher policy π_i^t, and let $\mathcal{N}(\mu_\theta(o), \sigma_\theta(o)) = \pi_\theta(o)$ denote the Gaussian distribution of the action output by policy π_θ on observation o. Given a batch of samples collected by the student policy π_s, $\{o_n, a_n, r_n, o'_n\}_{n=1}^B$, we use the following loss to train π_s:

$$L(\theta_s) = L_{SAC}(\theta_s) + \beta \sum_{i=1}^{N} L_{distill}(\theta_s, \theta_t^i), \tag{2.1}$$

where $L_{SAC}(\theta_s)$ is the standard SAC training loss, $L_{distill}$ is the policy distillation loss (described below), and β weighs the two terms. The policy distillation loss computes the distance between the student action distribution and the teacher action distribution, and thus the student learns to imitate the teacher's behaviors by minimizing such a loss. Specifically, we compute the Earth Mover's distance between the action distribution of the student and the teacher policy:

$$L_{distill}(\theta_s, \theta_t^i) = \sum_{n=1}^{B} \left(\mu_{\theta_s}(o_n) - \mu_{\theta_t^i}(o_n)\right)^2 + \left(\sqrt{\sigma_{\theta_s}(o_n)} - \sqrt{\sigma_{\theta_t^i}(o_n)}\right)^2 \tag{2.2}$$

The loss in Eq. 2.1 has also been used in prior work [94]; however, they used the KL divergence instead of Earth Mover's distance for the distillation loss. Our experiments indicate that earth-mover distance leads to significantly improved performance. We set $\beta = 0.01$ in our experiments.

Guided Domain Randomization Learning for Sim2real Transfer

We find that there is a huge difference between the simulated garment point cloud and the real-world garment point cloud, due to large variations in simulation vs. real garment geometries, and also since the simulator does not perfectly model the dynamics of garment deformations in the real world. To make the observation more aligned between simulation and the real world, and to make the policy robust to the observation difference, we add randomizations to the policy observation (described at the end of this section). Let o denote the original non-randomized point cloud observation, and let $\tilde{o}$ denote the randomized observation. Naively training the policy with randomized observation $\tilde{o}$ will usually fail or lead to degraded performance, since the randomizations make policy learning more difficult. To mitigate this issue, we propose "guided domain randomization", where we first train teacher policies π_i^t without any domain randomization; we then distill the teachers into a student policy π_s trained with domain randomization on the observation. Let θ_s denote the parameters of the observation randomized policy π_s. To train the student policy π_s, we run the student to collect a batch of data that stores both the randomized and non-randomized observation $\{\tilde{o}_n, o_n, a_n, r_n, \tilde{o}'_n, o'_n\}_{n=1}^{B}$, and train it as follow:

$$L(\theta_s) = \tilde{L}_{SAC}(\theta_s) + \beta \sum_{i=1}^{N} \tilde{L}_{distill}(\theta_s, \theta_t^i),$$

$$\tilde{L}_{distill}(\theta_s, \theta_t^i) = \sum_{n=1}^{B} \left(\mu_{\theta_s}(\tilde{o}_n) - \mu_{\theta_t^i}(o_n)\right)^2 + \left(\sqrt{\sigma_{\theta_s}(\tilde{o}_n)} - \sqrt{\sigma_{\theta_t^i}(o_n)}\right)^2,$$

where $\tilde{L}_{SAC}(\theta_s)$ is the SAC loss with running the policy π_s on the randomized observations, and $\tilde{L}_{distill}(\theta_s, \theta_t^i)$ is the loss of imitating the teacher policy π_i^t trained without domain randomization. Importantly, note that the student receives the randomized observation $\tilde{o}_n$ whereas the teacher receives the non-randomized observation o_n. For the observation randomization, we perform random cropping, dropping, erosion, and dilation on the cloth point cloud, and we add random noise to the robot gripper position. More details on the observation randomization can be found in Appendix A.1.2.

2.1.5 Simulation Experiments

Experimental Setup

We train our policies using Softgym [65] based on the NVIDIA Flex simulator. We use SMPL-X [84] to generate human meshes of different body sizes and arm poses. To generate random poses, the shoulder joint ϕ_1 is uniformly sampled from $[-20, 30]$, the inwards-outwards elbow joint ϕ_2 is uniformly sampled from $[-20, 20]$, and the upwards-downwards elbow joint ϕ_3 is uniformly sampled from $[-20, 30]$. We decompose the arm pose range Φ into $N = 27$ regions for policy distillation, by dividing each joint angle range into 3 intervals.

We randomly pick 5 garments for training from the Cloth3D dataset [9]: a hospital gown and 4 cardigans. The selected garments have different geometries such as varying sleeve lengths and widths. We randomly generate 50 human poses for each of the 27 pose sub-ranges, resulting in a total of $27 \cdot 50 \cdot 5 = 6750$ different configurations. Among the 50 poses for each sub-range, we use 45 poses for training and 5 poses for evaluation. At each training episode of SAC, we randomly sample one garment out of the five for training. The simulated dressing environment with the person holding different poses and with different garments is shown in Fig. 2.2. More details of the simulation experimental setup can be found in Appendix A.1.3.

Baselines and Ablations

We compare the following RL algorithms for learning the dressing policy from partial point cloud observations: **Dense Transformation policy (ours)** is our proposed method, which is described in Section 2.1.4. **Dense Transformation policy + latent Q function:** This baseline uses the same policy as our method. For the Q function, instead of concatenating the action to the input point cloud as an additional feature, this baseline first uses a PointNet++ to encode the point cloud observation into a latent vector, and then concatenates the action with the latent vector. An MLP then takes as input the concatenated latent vector and outputs a scalar Q value. **Direct Vector** uses a classification PointNet++ policy to encode the input point cloud to a single action vector. It uses a similar Q function network architecture as our method, where the action is concatenated to every point in the point cloud as an additional feature. **TD-MPC** [40], a state-of-the-art model based RL algorithm. **Deep Haptic MPC** [28], which learns a force prediction model based on end-effector measurements, and combines it with MPC to plan forward actions that minimizes the predicted force during dressing. The actions in MPC are sampled based on a "moving forward" heuristic. Note that this method does not use any visual information of the arm or the garment.

We also compare different ways of enabling the policy to generalize to diverse poses. **Policy Distillation** is our proposed method, as described in Section 2.1.4. **Policy Distillation (KL)** replaces the Earth Mover's distance in Equation (2.2) with KL divergence. **PCGrad** [121] is a multi-task learning algorithm that balances gradients from different tasks; in our case, a "task" is defined as training on a different pose sub-range. **No Distillation** trains a single policy on the entire pose range, without any distillation. **Heuristic Motion Planning** finds a collision-free robot end-effector path along the human arm based on some heuristically designed constraints. More implementation details of these baselines can be found in Appendix A.2.2.

In simulation, we use the *upper arm dressed ratio* as the evaluation metric. The

	Dense Transform (Ours)	Dense Transform Latent Q	Direct Vector	TD-MPC	Deep Haptic MPC
1	$\mathbf{0.68 \pm 0.05}$	0.55 ± 0.05	0.63 ± 0.07	0.04 ± 0.04	0.31 ± 0.08
2	$\mathbf{0.55 \pm 0.06}$	0.42 ± 0.08	0.52 ± 0.09	0.00 ± 0.00	0.11 ± 0.02
3	$\mathbf{0.75 \pm 0.04}$	0.68 ± 0.07	0.73 ± 0.02	0.03 ± 0.05	0.37 ± 0.02

Table 2.1: Upper arm dressed ratio of different policies on 3 pose sub-ranges. Results are averaged across 3 seeds.

ratio is computed between the dressed upper arm distance and the actual upper arm length: $\frac{||p_{int} - p^h_{elbow}||_2}{||p^h_{shoulder} - p^h_{elbow}||_2}$; see Section 2.1.4 and Fig. 2.3 for how these points are defined. This ratio is upper bounded by 1.

Effectiveness of Policy Architecture

We first compare the performance of different RL algorithms. We randomly select 3 different sub pose-ranges, and for each method, we train a separate policy on these 3 pose sub-ranges to compare their performances. We train each method for 1e6 steps, evaluate at each 10K steps, and report the maximum performance. The results are shown in Table 2.1 and averaged across 3 seeds. As shown, our proposed method Dense Transformation policy achieves the best performances on all 3 pose sub-regions, although Direct Vector has similar performance. Using a “Latent Q” architecture leads to much worse performance. We find that TD-MPC [40] does not work at all for this dressing task, possibly since the algorithm is originally designed and tested on image and state observations instead of point clouds, and also since it might be hard to learn a good latent dynamics model for deformable garments. Deep Haptic MPC also does not work well. The original paper tested only one arm pose and garment, with no visual information of the garment or human pose, potentially explaining its low performance in our setting with diverse arm poses and garments. Based on this result, we use Dense Transformation as our base RL algorithm for the following experiments.

Effectiveness of Policy Distillation on Generalization

We now compare the performance of different methods for learning a single policy on all 27 pose sub-ranges. We train all methods long enough until they converge (as different methods require different numbers of environment steps to converge), and

Policy Distillation (Ours)	Policy Distillation (KL)	PCGrad	No Distillation	Heuristic Motion Planning
0.68 $\pm$ 0.012	0.45 $\pm$ 0.010	0.37 $\pm$ 0.063	0.34 $\pm$ 0.10	0.32

Table 2.2: Upper arm dressed ratio of different ways to enable the policy to generalize to diverse poses. Results are averaged across 3 seeds. The Heuristic Motion Planning baseline does not have a standard deviation because there is no learning or randomness in this method.

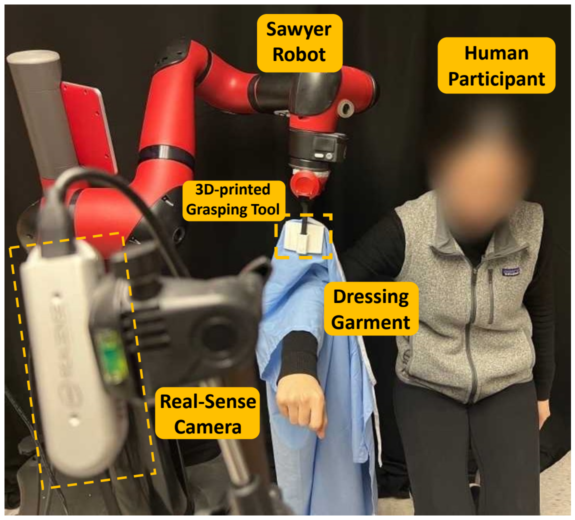


Figure 2.4: Real-world human study setup.

report the maximum achieved performance. Results are averaged across 3 seeds. As shown in Table 2.2, policy distillation with the Earth Mover's distance outperforms all other baselines. The baseline "No Distillation" and "PCGrad [121]" both perform poorly, showing the difficulty of directly learning a single policy across diverse pose ranges without distillation. The performance of the "Heuristic Motion Planning" baseline is also low. We find it performs well when there is no sharp bending at the shoulder and the elbow, and worse when the bends are sharp, aligning with findings in prior work [50]. Interestingly, we find that using Earth Mover's distance noticeably outperforms KL divergence, even though KL divergence seems to be the most common choice for policy distillation in the literature [83, 93, 94].

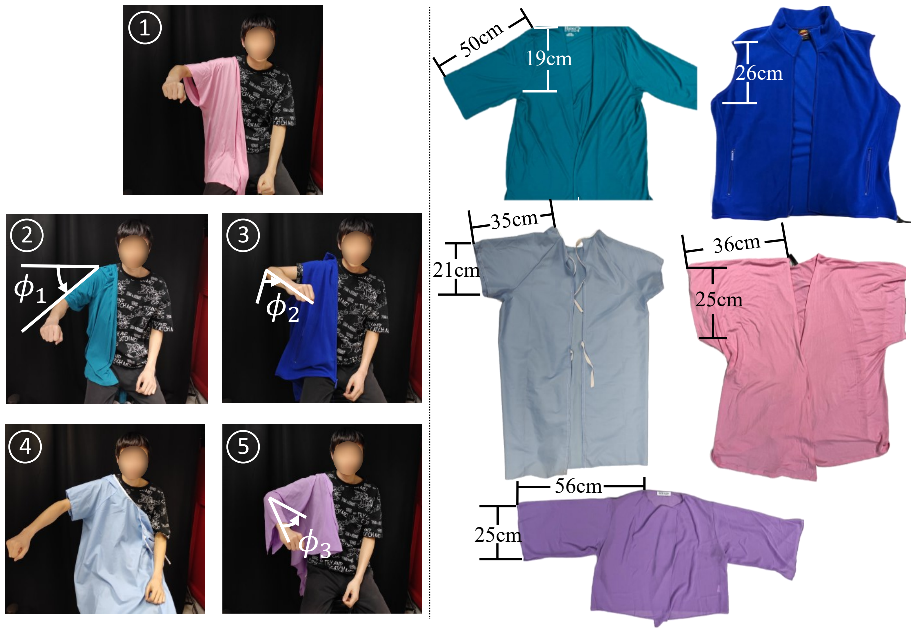


Figure 2.5: Arm poses and garments used for the human study.

2.1.6 Real-World Experiments and Human Study

We perform guided domain randomization (Section 2.1.4) to obtain a distilled policy that is robust for sim2real transfer, and deploy it in the real world, both for dressing a manikin and in a real-world human study. Our real-world experiments aim to answer the following questions: (1) Does our distilled policy outperform the baseline in the real world? (2) Does our policy generalize to different people with diverse poses and different garments?

Setup

Fig. 2.4 shows the real-world setup that we use for the human study, and Fig. 2.6 shows the manikin setup. Fig. 2.5 shows the arm poses and dressing garments we test. The poses we test in the real world span the pose ranges we train in simulation. We test 5 garments: a sleeveless, non-elastic vest; a short-sleeve, non-elastic hospital gown; a short-sleeve, elastic pink cardigan; a medium-length sleeve, narrow opening, elastic green cardigan; and a long-sleeve, non-elastic purple cardigan. Note that the real-world garments are not calibrated to the training garments in simulation; for example, the policy is never trained on a sleeveless vest in simulation. We use a

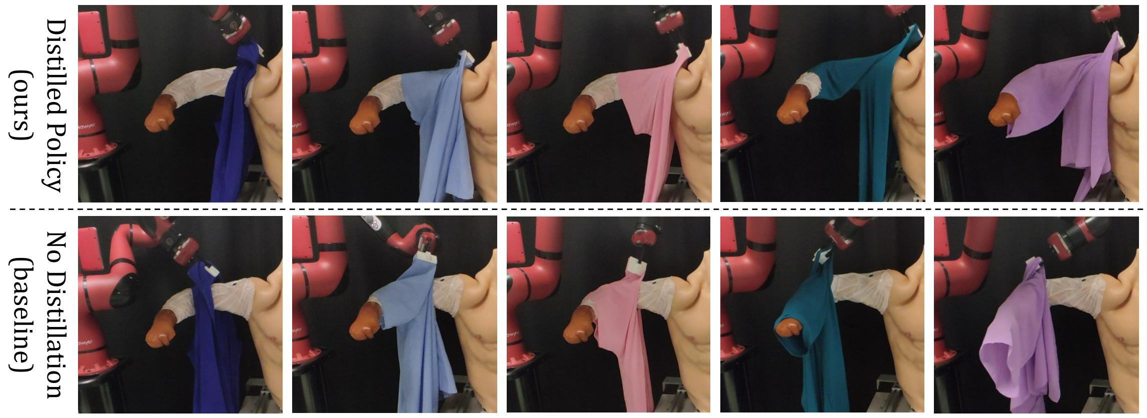


Figure 2.6: Final state comparison of the Distilled Policy (Ours, top) and No Distillation baseline (bottom) on the manikin.

						Total
Distilled Policy (Ours)	10/10	10/10	10/10	10/10	10/10	50/50
No Distillation	0/10	0/10	0/10	0/10	0/10	0/50

Table 2.3: Dressing success rate of the Distilled and No Distillation policies on 5 garments on the manikin.

Sawyer robot to execute the dressing task and an Intel RealSense D435i camera to capture the depth images.

We use the following quantitative metrics to measure the performance of a dressing trial. **Dressing Success:** We put a marker on the participant's shoulder, at the position that is 80% of the length up the upper arm. If the marker is covered by the garment at the end of the dressing trial, we consider this dressing trial is a success; otherwise it is not. **Upper Arm Dressed Ratio:** We compute the ratio between the "dressed upper arm distance" and the length of the participant's upper arm. The "dressed upper arm distance" is measured as the distance from the elbow to the intersection point of the garment and the participant's upper arm (see Fig. 2.3). **Whole Arm Dressed Ratio:** We compute the ratio between the "dressed whole arm distance" (the upper arm dressed distance + the forearm dressed distance) and the length of the participant's whole arm (upper arm length + forearm length).

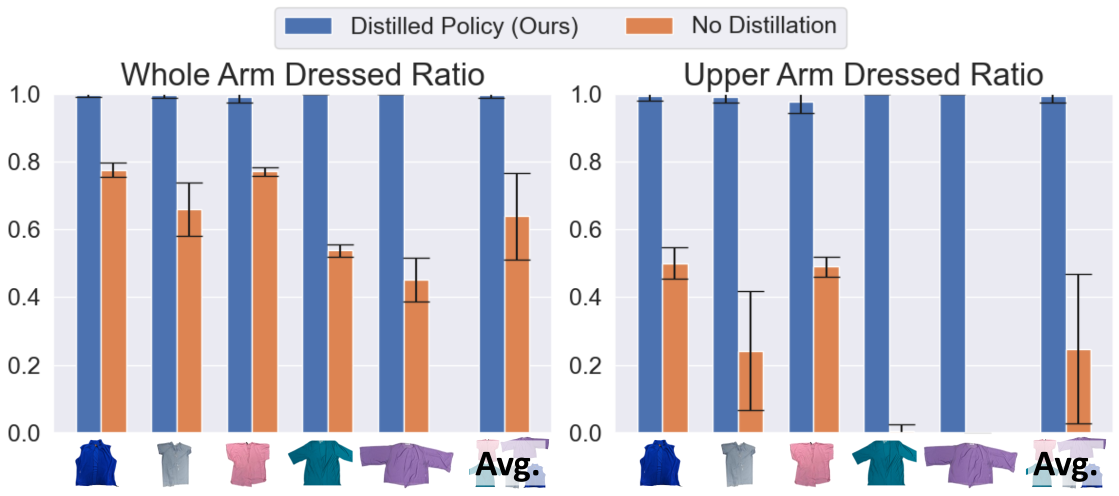


Figure 2.7: The whole arm (left) and upper arm (right) dressed ratio of the Distilled Policy and the No Distillation baseline on 5 garments on the manikin, averaged across 10 trials for each garment.

Comparison on a Manikin

In order to perform a strictly controlled comparison of the Distilled Policy and the No Distillation baseline, we first conduct experiments on a manikin shown in Fig. 2.6. The manikin has a fixed pose similar to pose 1 in Fig. 2.5. We conduct 10 dressing trials per garment (across 5 garments), resulting in 50 trials for each method. The numerical results are shown in Table 2.3 and Fig. 2.7. As shown, the Distilled Policy performs vastly better than the No Distillation baseline on all metrics, which resembles performance in simulation. The Distilled Policy successfully covered the marker on the shoulder every trial for all the garments. Fig. 2.6 shows the final state achieved by the Distilled Policy and the No Distillation baseline. For most of the trials, the No Distillation policy can only dress the forearm, and it has trouble correctly turning at the elbow to dress the upper arm.

Human Study Procedure

We recruit 17 participants in the human study, including 6 females and 11 males. The age of the participants ranges from 19 to 29. For each participant, we conduct 6 trials for each of the 5 garments, totaling 30 trials. Among the 6 trials per garment, we perform 5 trials using our distilled policy, corresponding to the 5 human poses

	Whole Arm Dressed Ratio	Upper Arm Dressed Ratio	Success Rate
Distilled Policy (Ours)	$\mathbf{0.86 \pm 0.19}$	$\mathbf{0.74 \pm 0.35}$	$\mathbf{0.66 \pm 0.47}$
No Distillation	0.63 ± 0.15	0.26 ± 0.27	0.01 ± 0.11

Table 2.4: Comparison of the Distilled Policy and the No Distillation baseline in a real-world human study. Results are averaged over 17 participants, on the 85 trials for which both methods are evaluated on the same poses and garments.

shown in Fig. 2.5. To compare against the baseline, we perform the remaining trial using the No Distillation baseline on a pose randomly chosen from the 5 poses. We test the No Distillation baseline with only one pose for each garment as we find that most participants experience arm soreness after 30 trials; thus it is impractical to test all poses with the baseline policy. Furthermore, since the No Distillation baseline shows poor performance on a static manikin as shown in Table 2.3, it is less likely that it would work on real humans with diverse arm shapes and poses. We randomize the test order of the garments, the poses, and the policy for each participant. During the study, the participant is not aware of which policy we are testing.

The procedure of each trial is as follows: We first show the participant the pose they should imitate, and they lift their right arm to maintain the pose. We then capture a point cloud of the participant's right arm. We move the Sawyer's end-effector, which is already holding the garment, to be positioned near the participant's hand. We then capture a point cloud of only the garment using color thresholding. The color-based segmentation of the garment could be replaced by training a garment segmentation network, such as fine-tuning an existing segmentation network with a small amount of data in our setting [52]. The point clouds of the human arm (recorded statically before the trial), the garment (color thresholded at each timestep), and the end-effector position (obtained using forward kinematics of the robot) are concatenated as input to the policy. The participant holds their arm steady throughout the trial. The trial terminates if any of the following conditions holds true: (1) if a maximum time step of 75 is reached, (2) if the participant's shoulder is covered by the garment, (3) if the policy is not making any progress in 15 consecutive time steps, and (4) if the participant wishes to have trial stopped. After the trial terminates, we measure dressing distances to compute our evaluation metrics. Finally, at the end of each

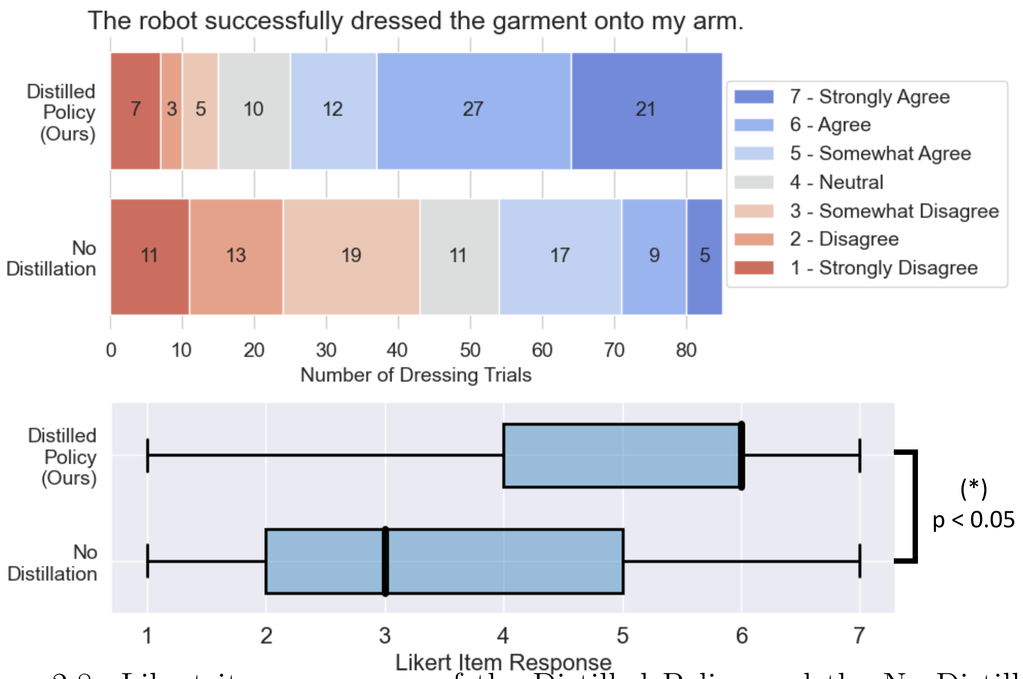


Figure 2.8: Likert item responses of the Distilled Policy and the No Distillation baseline, on the 85 dressing trials for which both methods are evaluated on the same poses and garments, shown as a full distribution (top) or box plot (bottom). We perform the Wilcoxon signed rank test on the Likert item response and find a statistically significant difference ($p < 0.05$) between our distilled policy and the baseline.

trial, we provide the participant with a single 7-point Likert item statement "The robot successfully dressed the garment onto my arm", which ranges from 1='Strongly Disagree' to 7='Strongly Agree'. More details on the human study procedure can be found in Appendix A.3.1.

Human Study Results and Analysis

We first compare the performance of the Distilled Policy against the No Distillation baseline in Table 2.5. The comparison is made on the 85 dressing trials for which both methods are evaluated on the same poses and garments. As shown, the Distilled Policy outperforms the No Distillation baseline by a large margin under all metrics. Fig. 2.8 shows the Likert item response distributions of our distilled policy versus the baseline. Overall, participants agreed at higher rates that the distilled policy successfully dressed the garment, as compared to the baseline. On average, our

	Whole Arm Dressed Ratio	Upper Arm Dressed Ratio	Success Rate
Distilled Policy (Ours)	$\mathbf{0.86 \pm 0.19}$	$\mathbf{0.74 \pm 0.35}$	$\mathbf{0.66 \pm 0.47}$
No Distillation	0.63 ± 0.15	0.26 ± 0.27	0.01 ± 0.11

Table 2.5: Comparison of the Distilled Policy and the No Distillation baseline in a real-world human study. Results are averaged over 17 participants, on the 85 trials for which both methods are evaluated on the same poses and garments.

Whole Arm Dressed Ratio	Upper Arm Dressed Ratio	Success Rate
0.86 ± 0.17	0.71 ± 0.34	0.57 ± 0.49

Table 2.6: Performance of our distilled policy, averaged over 17 participants and all 425 dressing trials (not just the 85 trial subset in Table 2.5).

distilled policy achieves a median response of 6.0, meaning that the participants "Agree" that the robot successfully dressed the garment onto their arm, while the No Distillation baseline achieves a median response of 3.0, meaning the participants "Somewhat disagree." Let d be the difference between the Distilled Policy's Likert item response and the No Distillation baseline's Likert item response. We perform the Wilcoxon signed rank test to test if the distribution of d is stochastically greater than a distribution symmetric about zero. We obtain a p-value of $p = 0.03125$; hence we find a statistically significant difference ($p < 0.05$) between our distilled policy and the baseline, and the median of the difference is positive.

We now analyze our policy's performance over all 425 dressing trials from the 17 participants. Table 2.6 shows the Distilled Policy's performance under all metrics. On average, our policy is able to dress 86% of the participant's whole arm, and 71% of the participant's upper arm, achieving a final dressing state similar to the 5th sub-image in Fig. 2.10. Fig. 2.13 shows snapshots of the dressing trajectories of our policy. In terms of Likert item responses, as shown in Fig. 2.9, our distilled policy achieves a median response of 5.0 ("Somewhat Agree" that the robot successfully dressed the garment onto the arm). Note that the metrics and Likert item responses are different from those in Table 2.5 and Fig. 2.8 because here the numbers are averaged across all dressing trials, as opposed to the 85 trial subset used for the baseline comparison.

Fig. 2.11 shows the performance of our method across different garments. It

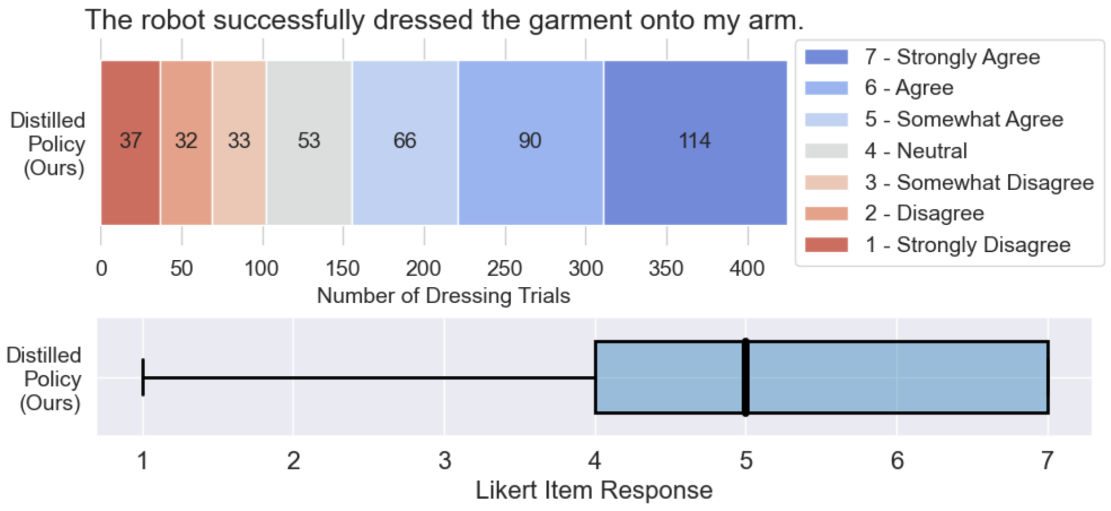


Figure 2.9: (Top) Full distribution and (Bottom) box plot of the Likert item responses of the Distilled Policy on all 425 dressing trials (not just the 85 trial subset shown in Fig. 2.8).

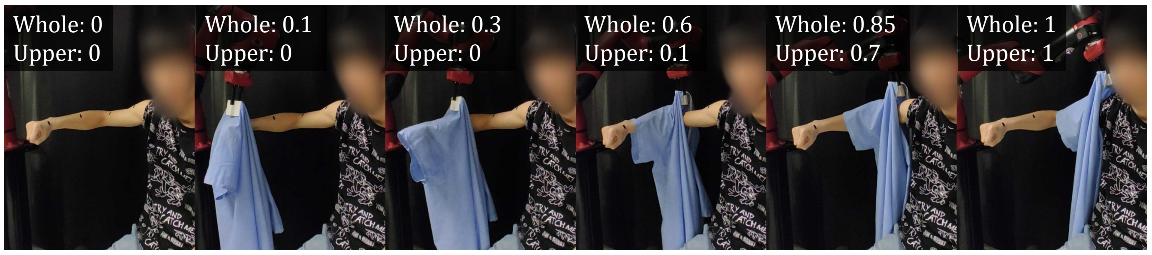


Figure 2.10: An illustration of the dressed ratios during a trial.

is interesting to see a consistent ordering of the garments under all metrics. Such ordering aligns with human intuitions based on the garment geometry & materials (see Fig. 2.5): the vest is sleeveless and thus the easiest to perceive and dress (even though the sleeveless vest has never been trained in simulation). The green and purple cardigans, however, have longer sleeves and are harder in terms of both perception and dressing. Meanwhile, although the pink cardigan and the gown have similar sleeve length, the material of the pink cardigan is much more elastic than the hospital gown. This provides some allowance in the end-effector trajectories during the dressing process, leading to better performance. This also holds true for the purple versus green cardigan – although the opening of the purple cardigan is larger, it is less elastic, which results in the garment getting caught on the participants' arm more frequently, leading to a failure.

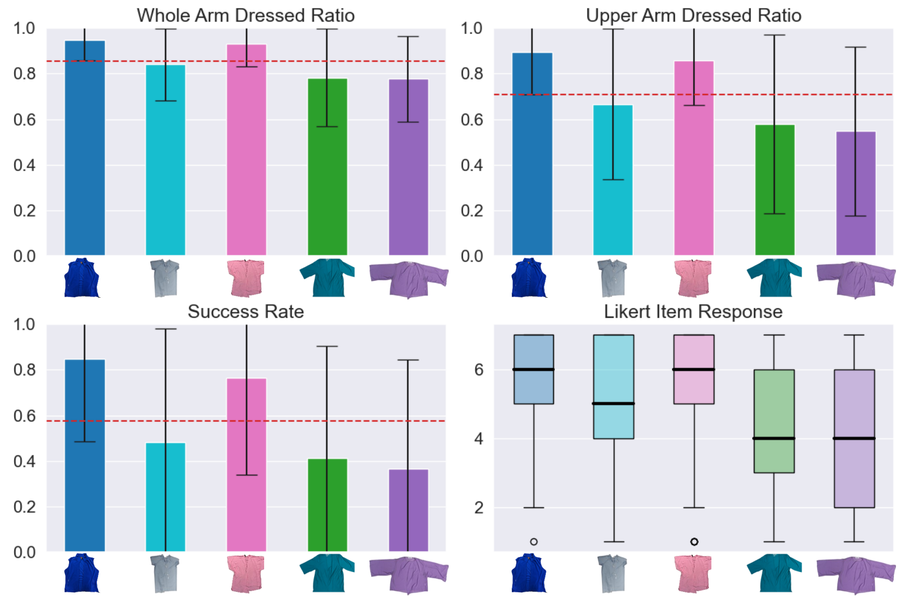


Figure 2.11: Performance of our distilled policy on different garments. The dashed red line shows the average over all garments.

Finally, we analyze the performance of our distilled policy on different poses in Fig. A.4. We notice that some poses are harder than others. For example, the fifth pose with sharp bends at the shoulder and elbow has the lowest performance. On the other hand, the first pose, which does not have a sharp bend at either shoulder nor elbow, achieves the highest performance. Such a difference is likely due to the fact that sharp bends require more complex end-effector trajectories, and the garment is also more likely to get caught at sharp bends. We also note that although the first pose in the human study is similar to the pose of the manikin in Fig. 2.6, the success rate on this pose in the human study is lower than the one achieved on the manikin shown in Table 2.3. We speculate that such a difference could be due to the diversity of the arm shape and sizes in the human study, as some participants' arm sizes and shapes might be out of our training distribution, resulting in lower performance. Another reason could be that compared to the manikin, real people unconsciously move their arms during the dressing trial, which violates our assumption that the human holds a static arm pose.

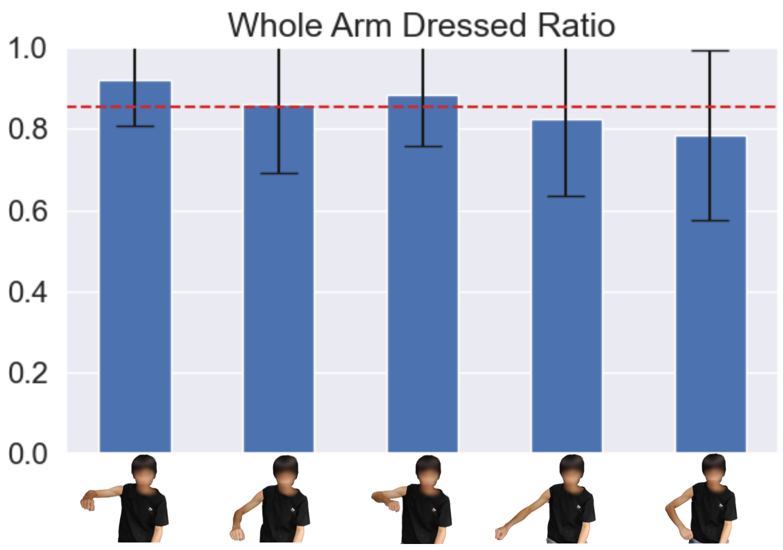


Figure 2.12: Whole arm dressed ratio of our distilled policy on different poses. The dashed red line represents the average over all poses.

Failure Cases Analysis

Visual examples of failure cases can be found in our project webpage and also in Appendix A.3.3. The first failure case is that the policy gets stuck and cannot output actions that make further progress for the task, e.g., the policy oscillates between moving up and down and does not move forward. This might be due to that the arm pose of the participant is out of the training distribution, or other aspects of the observation cause a sim2real gap. Another failure case is the cloth getting caught on the arm. This usually happens when the participant unintentionally moves their arm too much in the dressing process, the policy actions move the gripper too high above the participant's arm, or the policy actions turn too early at the elbow. Due to limited fidelity of the simulator, the garment in simulation is more elastic and can still be dressed even if the robot is pulling high above the arm. However, the garments we test in the real world are less elastic than the simulated ones and they experience greater friction and get caught more easily. We believe that fine-tuning the policy in the real world can help address both failure cases. Since it is difficult to visually detect if a garment is undergoing high friction or has gotten caught on the body, we believe incorporating force-torque sensing can help alleviate these issues. We leave both as future work.

2.1.7 Out-of-distribution Evaluation and Generalization of the System

We conducted a preliminary out of distribution evaluation of our system by relaxing the static arm assumption, i.e., the participants can move their arm during the dressing process. With experiments in both simulation and real world, we observe our system to be robust to small arm movements. We first evaluate how our system performs if the participants change their shoulder or elbow joint angles after we capture the initial arm point cloud. The simulation experiments are conducted on 1 pose sub-region, and the results show that our system is robust to 8.6 degrees of change in shoulder and elbow joint angles (averaged across 3 types of joint angle changes and 5 garments) while maintaining 75% of the original performance. Please refer to Appendix A.4 for detailed experimental results. We also conducted 4 real-world dressing trials with one participant changing their joint angles, including lowering down the shoulder joint for 5 degrees, lowering down the elbow joint for 5 degrees, bending the elbow joint inwards for 5 degrees, and bending the elbow joint inwards for 10 degrees. Our system succeeds in the first 3 trials, and fails for the last trial since the elbow joint angle change is too large. We also evaluate our system with one participant performing constant arm motions during the dressing process in the real world. The participant is asked to perform 4 different kinds of arm motions during dressing, including constantly moving their forearm horizontally, constantly moving their forearm in a spherical motion, constantly moving their forearm up and down, and constantly moving their shoulder up and down. The maximum displacement of the arm during the motion was ± 10 centimeters. Our system succeeds in all 4 kinds of arm motions. Please see our website for videos of these real-world evaluations.

We also evaluate the generalization of the system towards dual-arm dressing. With the same single-arm dressing assumptions (static arm pose and robot already grasping the garment), our system generalizes to dual-arm dressing. The primary change is to control two robotic arms, one for each sleeve of the garment, which our Dense Transformation policy can handle well by extracting actions corresponding to both of the robot gripper points. We verified this in simulation with preliminary experiments where we successfully train policies to perform dual arm dressing of a hospital gown and a cardigan on a fixed pose (see visualizations on our project website). We note that dressing over a person's head, such as with a t-shirt, is more complex, due to the need for more dexterous trajectories and awareness of safety considerations. We leave such an extension to future work.

2.1.8 Conclusion

In this work, we develop a robot-assisted dressing system that is able to dress diverse garments on people with diverse poses from partial point cloud observations, based on a learned policy. We show that with careful design of the policy architecture, reinforcement learning (RL) can be used to learn effective policies with partial point cloud observations that work well for dressing diverse garments. We further leverage policy distillation to combine multiple policies trained on different ranges of human arm poses into a single policy that works over a wide variety of different poses. We propose guided domain randomization for effective and robust sim2real transfer. We perform comprehensive real-world evaluations of our system on a manikin, and in a human study with 17 participants of varying body size, poses, and dressed garments. On average, our system is able to dress 86% of the length of the participants' whole arm, and 71% of the length of the participants' upper arm across 425 dressing trials. We hope this work will serve as a foundation for future research to develop more robust and effective dressing systems.

2.2 Force-Constrained Visual Policy

2.2.1 Introduction

Dressing is a crucial activity for individuals with disabilities or limited mobility to receive assistance with. Recent studies [41] estimate that 92% of all residents in nursing facilities and at-home care patients require assistance with dressing. Robot-assisted dressing has emerged as a potential solution to these challenges [35, 50, 107, 128], which could be used to enhance the quality of life of people with physical disabilities. In this work, we demonstrate a new learning-based method for combining vision and force sensing modalities towards a safe and comfortable assistive dressing system.

Robot-assisted dressing comes with several challenges. First, robotic manipulation of deformable garments is challenging due to the lack of a compact state space representation, complex cloth dynamics, and self-occlusions of clothing. Moreover, during robot-assisted dressing, the robot works in proximity to the human and has direct physical contact with the human body. Undesired motions performed by the robot that strain the garment or cause accidental collisions with a person could apply large forces to the human body and pose discomfort and potential safety risks.

Prior work in cloth manipulation and robot-assisted dressing has demonstrated the use of vision [34, 107] and force [28] modalities separately to make control decisions. Yet, there is a clear advantage to leveraging both modalities simultaneously [130]. Visual sensing is useful to observe the garment and human arm to infer a reasonable dressing path, and force sensing is needed to ensure safety and comfort during the dressing process. In this context, simulation can be used to collect large amounts of data to train a control policy that can generalize across diverse people, body poses, and garments. Prior work [107] has demonstrated the ability to transfer point cloud-based assistive dressing policies from simulation to the real world; however, most simulators do not provide sufficiently accurate robot force sensing when manipulating deformable garments around human bodies, which limits the transfer of force-based models from simulation to the real world. This reality gap necessitates learning from force measurements directly in the real world. The question that we explore in this paper is how to combine a visual policy for assistive dressing trained in simulation with force sensing data that is only available in the real world.

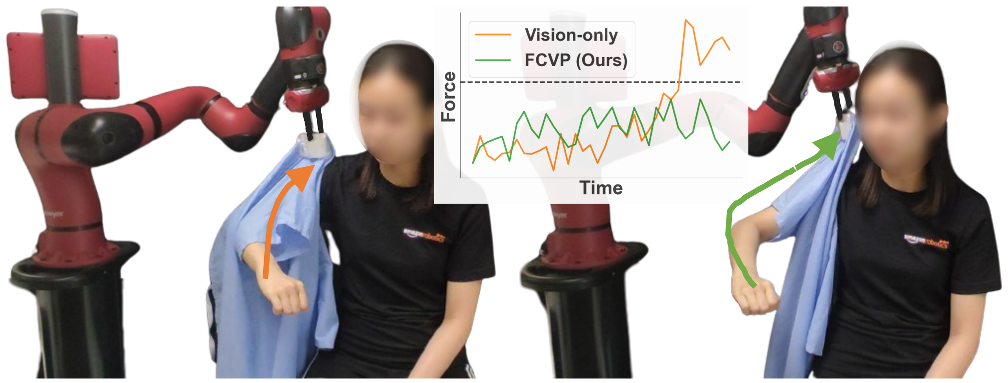


Figure 2.13: Our method learns a force dynamics model in the real world to constrain the vision-based policy trained in simulation (right), preventing high force from being applied to the person (left).

In this paper, we propose a new method for the task of assistive dressing, named Force-Constrained Visual Policy (FCVP), shown in Figure 2.13. Our method elegantly handles the case in which only the visual modality (using point clouds) can be simulated sufficiently accurate to be transferred to the real world, but the force modality cannot. Our key idea is to use a vision-based policy trained in simulation to propose actions, and then to use a force-based dynamics model trained in the real world to filter out unsafe actions. We comprehensively evaluate our method with a real-world human study with 10 participants and 240 trials, demonstrating its strong effectiveness.

In summary, we make the following contributions:

- We propose a new method for multi-modal learning when one sensor cannot be well-modeled in simulation. Our method, Force-Constrained Visual Policy (FCVP), combines a vision-based policy trained in simulation with a force-based dynamics model trained in the real world.
- We evaluate our method rigorously in simulation and also perform a real-world human study with 10 participants and 240 dressing trials to quantify the real-world practicality and efficacy of the proposed method and system. These experiments demonstrate that our method leads to a safe and comfortable assistive dressing system with higher dressing performance by ensuring low forces are exerted to the human.

2.2.2 Related Work

Multi-Modal Learning for Robotic Manipulation

Recently, there has been an increasing number of works studying multi-modal learning, which combines modalities such as vision and force [130], vision and touch [16, 58, 100, 101, 109, 111], and vision and audio [25, 62, 120], with applications in grasping [16, 109], object manipulation [8, 62, 101, 111], assistive tasks [100, 129, 130], and many more. Most prior works on multimodal learning focus on how to design a more effective policy network architecture that takes all modalities as input. Most of them directly train in the real world [16, 25, 58, 62, 100], via imitation learning [25, 62, 100], supervised learning [16], or self-supervised learning followed by reinforcement learning [58], which all require a large amount of human-collected datasets or robot trials. In contrast, our approach trains the vision-based policy in simulation. Some approaches train a multimodal policy in simulation and perform sim2real transfer, such as vision with contact points [109], and vision with rigid-body force sensing [8]. These approaches assume that all of the modalities can be accurately simulated; in contrast, our method trains a force-based dynamics model directly in the real world, without assuming that the force modality can be accurately simulated for modeling cloth-human force interactions.

In contrast to these prior works which employ a single policy network that handles both modalities, our proposed approach combines a vision-based policy trained in simulation and a force dynamics model trained in the real-world via solving a constrained optimization problem.

Safe Reinforcement Learning

The objective of developing a safe robot-assisted dressing system can be formulated as a safe reinforcement learning problem [36, 69], where the reward is to dress the person, and the safety constraint is that the amount of force applied to the person should be below a threshold. However, there could be some issues of directly applying safe RL algorithms to our problem setting: if we train the safe RL policy in simulation using both vision and force, then it will be difficult to transfer to the real world since the simulator does not provide accurate enough force simulation with deformable cloth. Our method avoids this issue by training a vision-based policy in simulation and a force-based dynamics model in the real world.

2.2.3 Problem Statement and Assumptions

As shown in Figure 2.13, we study the task of single-arm dressing assistance, where the goal is to dress the sleeve of a garment onto a person's arm. Single-arm assistive dressing is a fundamental skill of full upper-body dressing assistance for individuals with motor impairments. We want to achieve safe dressing assistance by ensuring that low forces are exerted to the human during the dressing process.

Formally, let $f_t \in \mathcal{R}$ be the amount of force the garment exerts onto the human body at time step t (which we approximate through force sensing at the robot's end-effector). We want to develop a safe robot-assisted dressing system that can pull the garment to cover the human arm and shoulder, while maintaining the force applied to the human to be below a threshold τ, i.e., $f_t < \tau$, $\forall t \in [1, T]$, where T is the horizon of the task. We assume the person holds their arm static during the dressing process, and that the robot has already grasped the opening of the garment shoulder in preparation for dressing. While not the focus of this paper, prior works have proposed methods for grasping garments [127, 128] and adapting to human motion during dressing assistance [30], which could be integrated into our work.

2.2.4 Background - Vision-based Policy Learning in Simulation

Our method leverages a vision-based policy π^v trained in simulation from prior work [107]. We describe the core training procedure here, and refer to [107] for more details. The dressing task is formulated as a Partially Observable Markovian Decision Process (POMDP) and is solved via reinforcement learning. The core components of the POMDP are defined as follows:

Observation Space O: The policy observation is the segmented point cloud of the scene, which consists of the garment point cloud P^g and the human arm point cloud P^h. As we assume the human to be static, we can obtain the static arm point cloud P^h before the garment occludes the arm and use it during the whole dressing process;

thus our input includes the full arm even when the garment occludes it. A single point P^r at the location of the robot's end-effector is added to the observation. The full observation o is the concatenation of these three point clouds: $o = [P^g; P^h; P^r]$ (see Figure 2.14 for a visualization). The feature for each point in o is a one-hot encoding indicating which object the point belongs to, i.e., the garment, the human arm, or the end-effector.

Action Space A: the action $a \in A$ is defined as the delta transformation for the robot end-effector. It is a 6D vector, where the first 3 elements denote the delta translation, and the second 3 denote the delta rotation represented as axis angle.

Reward r: The reward r consists of a term that measures the task progress, which is the dressed distance of the garment along the human arm, with additional auxiliary reward terms to prevent the gripper from moving too close to the person. The full detailed reward function is the same as in Wang et al. [107], and can be found on our project website.

SAC [39] is used as the underlying RL algorithm for training the vision-based policy, and a segmentation-type PointNet++ [88] as the policy architecture (see [107] for details). As in prior work [107], the vision-based policy is trained in simulation on many variations of body shapes, arm poses, and garments, and can be transferred to a real world manipulator. However, the actions inferred by the vision-based policy may exert high forces onto people when deployed in the real world. Our method handles this by further learning a force dynamics model in the real world.

2.2.5 Method

Method Overview: As shown in Figure 2.14, our system is comprised of two parts. First, we leverage a vision-based policy from prior work which is trained in simulation using reinforcement learning (as described in section 2.2.4). By using simulation, we are able to easily collect a large amount of data and train a single policy that can generalize to many variations of human arm poses, body shapes, and garments. The vast amount of data needed makes it prohibitive to train the vision-based policy directly in the real world. To ensure safe assistive dressing, we learn a force dynamics model which predicts the future forces applied to the human. The force dynamics model is trained in the real world, due to the fact that many simulators do not

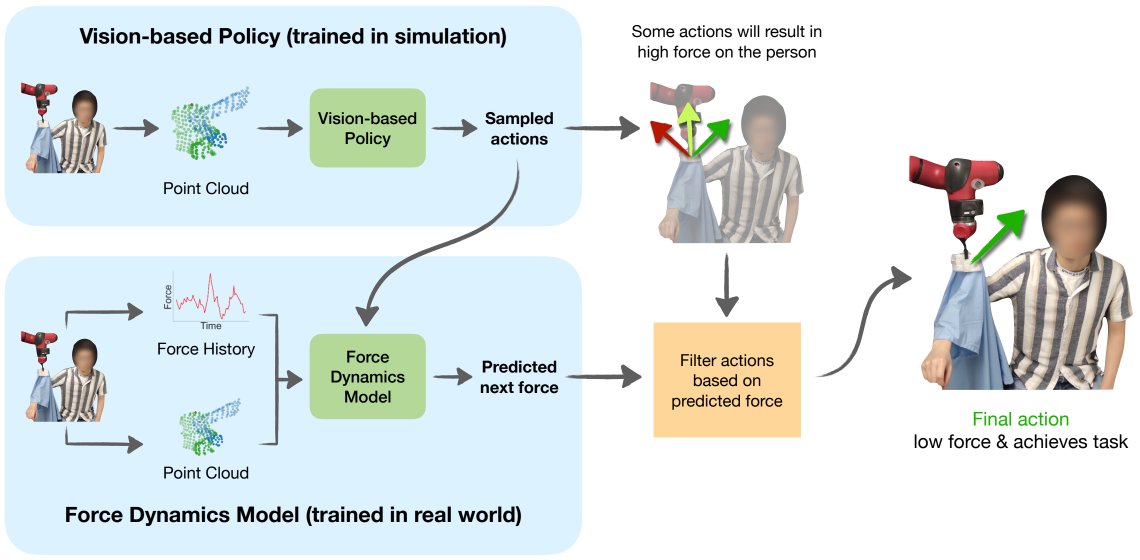

Vision-based Policy (trained in simulation)
Point Cloud
Vision-based Policy
Sampled actions
Some actions will result in high force on the person
Force
Time
Force History
Point Cloud
Force Dynamics Model
Predicted next force
Filter actions based on predicted force
Final action
low force & achieves task
Force Dynamics Model (trained in real world)

Figure 2.14: Our system combines a vision-based policy and a force dynamics model to achieve safe robot-assisted dressing. As most simulators provide sufficiently accurate simulation of point clouds yet not the force modality for sim2real transfer, the vision-based policy is trained with a large amount of data in simulation, and the force dynamics model is trained with a small amount of data in the real world. At test time, the vision-based policy proposes action samples that progress the dressing task. The force dynamics model predicts the future forces of these sampled actions, and the predictions are used to filter actions that are unsafe, i.e., those applying too much force to the human. The final chosen action is safe with low force and achieves the task.

provide sufficiently accurate force simulation for deformables manipulated around the human body. At test time, the final robot action is inferred by solving a constrained optimization problem that combines the vision-based policy and the force dynamics model.

Force dynamics model learning in the real world

As previously mentioned, force simulation of deformable garments is not sufficiently accurate to transfer from simulation to the real world. Even after system identification, it is challenging to accurately estimate all the local forces caused by cloth deformation and stretch when interacting with a human or other objects in the environment. Therefore, we aim to directly learn the force-based model in the real world.

Specifically, we learn a force dynamics model $d_\psi(o_t, F_t, a_t)$, which takes as input the current point cloud observation o_t as described in section 2.2.4, the past N steps of forces $F_t = [\tilde{f}_t, ..., \tilde{f}_{t-N+1}]$ (where $\tilde{f} \in \mathcal{R}^3$ is a three-dimensional force vector), and the robot action a_t. The force dynamics model predicts the future amount of force $\hat{f}_{t+1} \in \mathcal{R}$ the human will experience due to robot action a_t. To predict the future force, the force dynamics model uses a PointNet++ [87] encoder to encode the point cloud observation o_t into a latent vector. This latent vector is then concatenated with the force history F_t and the action a_t. Another MLP then receives as input the concatenated vector and outputs the predicted force $\hat{f}_{t+1}$ in the next timestep.

We collect training data for the force dynamics model in the real world using the vision-based policy π^v. To train the force dynamics model on a wider range of action distributions, at each time step, the action a_t is sampled as following: with probability p, a_t is uniformly sampled from $[-1, 1]^{|A|}$ (where $|A|$ is the dimension of the action space), and with probability $1 - p$, it is sampled from $a_t \sim \pi^v(o_t)$. The force dynamics model (including the PointNet++ encoder and the MLP) is trained using the MSE loss to predict the future force: $L(\psi) = ||d_\psi(o_t, F_t, a_t) - f_{t+1}||_2^2$, where ψ denotes the parameters for the force dynamics model.

Force-Constrained Vision Policy

At test time, we combine the trained vision policy π^v and the force dynamics model d_ψ to infer the action by solving the following constrained optimization problem:

$$\arg\max_{a} \quad \pi^v(a|o) \quad \text{subject to} \quad d_\psi(o, F, a) \leq \tau \tag{2.3}$$

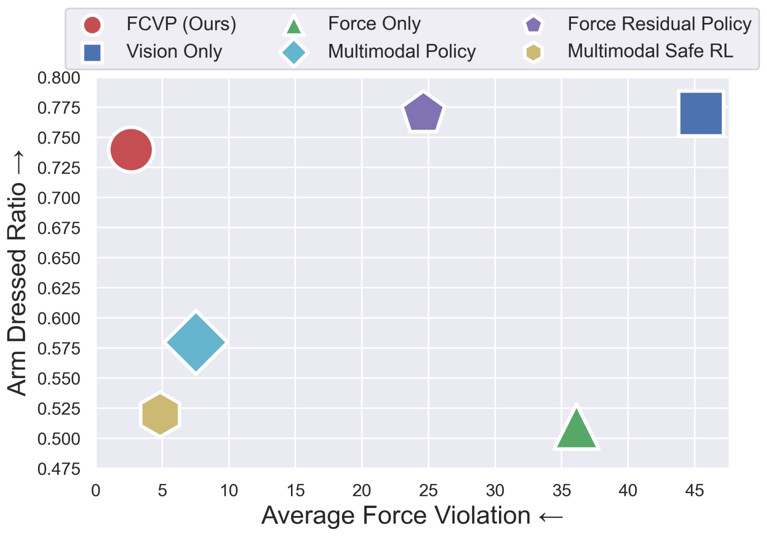
where τ is a user-chosen force threshold. There are several optimization algorithms for solving such a constrained optimization problem, such as Lagrangian method, active-set method, interior-point method, random shooting method, and more. As the functions involved in our constrained optimization problem are represented using neural networks, we use the random shooting method to solve the optimization problem due to its simplicity and low computational cost. The random shooting method works as follows: we sample a set of actions, filter out any actions whose predicted forces are above the threshold τ, and among the actions whose predictions are below the threshold, we execute the action with the highest probability under the vision-based policy π^v. We use the same action sampling distribution as during training (i.e. a mix of actions from the vision policy π^v and actions from a uniform distribution). If there is no action whose predicted force is below the threshold, we execute the action with the lowest predicted force.

2.2.6 Experiments

We conduct both simulation and real-world experiments to evaluate our method. We first perform sim2sim transfer experiments to compare FCVP with other multimodal learning methods (Section 2.2.6). We also perform real-world human studies (Carnegie Mellon University IRB Approval under 2022.00000156) to evaluate the effectiveness of the robot-assisted dressing system (Section 2.2.6).

Sim2sim Transfer Experiments

Setup: In order to test our method in a controlled setting, we create two simulation environments with different simulation parameters (the detailed parameters can be found on our project website); we treat one of them as simulation (sim A), and the other as an approximation to the real world (sim B). The force readings between these



	Arm Dressed Ratio ↑	Average Force Violation ↓	# Training trajectories in sim B ↓
FCVP (Ours)	0.74 ± 0.29	**2.65**	225
Vision Only	$\mathbf{0.77 \pm 0.33}$	45.45	**0**
Force Only	0.51 ± 0.28	36.10	225
Multimodal Policy	0.58 ± 0.42	7.51	7987
Force Residual Policy	$\mathbf{0.77 \pm 0.25}$	24.60	8190
Multimodal Safe RL	0.52 ± 0.40	4.83	8278

Figure 2.15: (Top) Among all compared methods, FCVP achieves the best trade-off between the arm dressed ratio and the force violation amount. (Bottom) The detailed quantitative results for each method, as well as the number of training trajectories required for convergence in sim B.

two environments are different due to the differences in the simulation parameters, approximating the sim2real gap. We use NVIDIA FleX [72] wrapped in SoftGym [65] as the simulator.

We use SMPL-X [84] to generate human meshes with distinct body shapes, sizes, and arm poses. Specifically, we generate 4 different arm pose regions; the arm poses within each region are similar to each other with small variations, and the arm poses are very different across regions (see figures of the arm pose regions on our project website). For each region, we generate 45 human meshes of distinct body shapes and sizes. All methods are evaluated on each arm pose region, and then the results are averaged across the arm pose regions.

We use the same garments as in Wang et al. [107]: a common hospital gown and 4 cardigans from the Cloth3D dataset [9]. In simulation, we set the force threshold τ

to be 40 units (note due to simulation inaccuracies, this unit does not correspond to Newtons in the real world.)

We compare the following methods:

FCVP (Ours): For our method FCVP, we learn a vision-based policy in sim A. The force dynamics model is learned in sim B. We collect one trajectory per human mesh and garment, resulting in a total of 225 trajectories in each arm pose region for training the force dynamics model in sim B. We use the past 5 steps of force measurements as input to the force dynamics model.

Vision Only directly transfers the vision-based policy trained in sim A to sim B, without any fine-tuning nor using the force information.

Force Only uses only the force dynamics model trained in sim B, without using the vision-based policy pretrained in sim A. The actions are planned by minimizing the predicted force, and are heuristically sampled within a forward task progression cone, similar to the method used in [28].

Multimodal Policy: We first pretrain a policy that takes as input the vision and the force modality in sim A using RL, and then we fine-tune it with vision and force in sim B. This is the most standard approach for multimodal learning.

Force Residual Policy: We first pretrain a policy that only uses the vision modality in sim A, and then we train a residual policy on top of this pretrained vision-based policy using both vision and force in sim B. The residual policy is trained to output a delta action, which is added to the action from the pretrained vision-based policy.

Multimodal Safe RL: This is similar to Multimodal Policy, but instead of using RL to pretrain or fine-tune the policy, we use safe-RL. We use SAC-Lagrangian [69] as the training algorithm. Specifically, we first pretrain a policy that takes as input the vision and the force modality in sim A using SAC-Lagrangian, and then we fine-tune it with vision and force in sim B with SAC-Lagrangian.

For training the multimodal policy and the force residual policy, to encourage the policy to exert low force to the human, we add an additional penalty term to the reward when the force is above the threshold. For the multimodal safe RL baseline, the force is treated as the cost for SAC-Lagrangian.

All methods are evaluated in sim B, to demonstrate the ability of each method to transfer to new dynamics. We provide additional details of these baselines on our project website. We train all multimodal baselines till convergence, which usually require a magnitude more data than FCVP.

Evaluation Metrics: The first evaluation metric is the *Arm Dressed Ratio*, which is the ratio between the dressed arm distance and the real arm length. A ratio of 1 means that the arm is fully dressed; 0 means that the arm is not dressed at all. This metric is used to measure the dressing performance and has been used in related work [107]. The second metric is the *Average Force Violation*, which is computed as $\frac{1}{T}\sum_{t=1}^{T}\max(0, f_t - \tau)$, in which f_t is the measured force at time step t and τ is the maximal force threshold.

Results: Figure 2.15 presents the performances of all methods. As shown on the left subplot, FCVP achieves the best trade-off between the dressed ratio and the force violation amount: it achieves the lowest force violation, and the third highest arm dressed ratio. All other baselines either have large force violations (Force Residual Policy, Vision Only, Force Only), or perform poorly on the dressing task, as indicated by a low Arm Dressed Ratio (Multimodal policy, Multimodal Safe RL, Force only). As shown in the right table of Figure 2.15, most other multimodal learning baselines require much more training data in sim B, which serves as a proxy of the real world. This is because they require policy fine-tuning via RL in sim B. In contrast, FCVP learns a dynamics model in sim B, which is a supervised learning problem, thus being much more sample-efficient to learn. Overall, these results show that both vision and force information are needed to achieve high dressing performance while being safe; further, our approach of incorporating vision and force in FCVP not only achieves better performance, but also requires much less training data in the "real world" (sim B) than other multimodal learning methods. Box plots of the force distributions for all methods are provided on our project website for further visualization.

Ablation study. We first investigate how the number of past force measurements, denoted as N, affects the performance of FCVP. We test $N = 3, 5, 7$ and find that a larger N leads to a decrease in force violations but also a reduced dressed ratio. We use $N = 5$, which achieves a good trade off between these two objectives. We also study the impact of including past actions as part of the input for the force dynamics model. We test including $0, 1, 3, 5$ steps of past actions, and find the performance varies minimally across these different lengths. The best performance is achieved when the model does not include past actions as part of its input. Please refer to our project website for further details on these ablation studies.

Real-World Human Study

Experimental Setup: Figure 2.16 shows the setup for our real-world human study. We use the Sawyer robot for executing the dressing task, and measured the force at the Sawyer robot's end-effector (wrist) using Sawyer's built-in force sensing. The robot movement (action) is the delta translation and rotation of the end-effector, and is executed using the Sawyer's built-in IK solver and an impedance controller. We use a single Intel RealSense D435i camera to capture the point cloud observations of the scene. We compare our method with the following two baselines:

Vision Only [107] chooses the action with the highest probability under the vision-based policy.

Vision with Random Actions samples the action from the vision-based policy with probability p, and uniformly randomly from $[-1, 1]^{|A|}$ with probability $1 - p$, where $|A|$ is the dimension of the action space (6 in our case).

We compare to the "Vision with Random Actions" baseline as it is also used for collecting data for training the force dynamics model, as mentioned in section 2.2.5. We set $p = 0.1$ and the force threshold τ to be 5.4 Newtons in our experiments, which is an empirical value based on the force distributions of successful and safe dressing trials, and we think this value would be comfortable for the participants during the dressing process. We do not compare to the *Force Only* baseline because it performs poorly in terms of the dressed ratios even in simulation (see table in Figure 2.15). We also do not compare to other multimodal learning baselines, as they all require extensive amounts of training data to converge (as shown in the sim2sim transfer experiments in Figure 2.15), which is prohibitive to collect in the real world.

In addition to the evaluation metrics described in section 2.2.6, we present participants with 7-point Likert items that range from 1='Strongly Disagree' to 7='Strongly Agree' with the following statements: 1. "The robot successfully dressed the garment onto my arm"; 2. "The force the robot applied to me during dressing was appropriate"; and 3. "The dressing process was comfortable for me."

Human Study Procedure: We recruit 10 participants, comprising 4 males, 4 females, 1 non-binary individual, and 1 participant who chooses not to disclose their

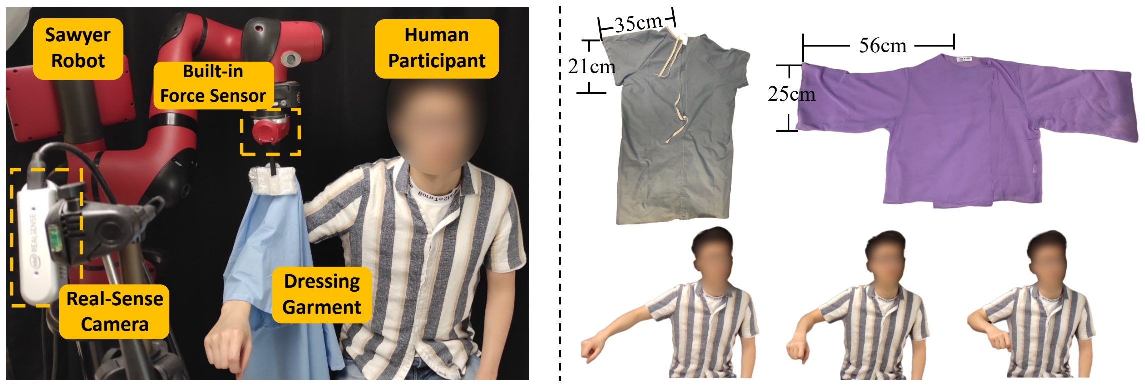


Figure 2.16: Left: Human study setup. Right: Poses and Garments that we test in the human study.

gender identity. The age of the participants spans from 19 to 36. We test each participants with two garments: a short-sleeve hospital gown and a long-sleeve purple cardigan. These two garments differ in geometry (see Fig 2.16 for sleeve lengths and widths), elasticity and roughness (the purple cardigan is more elastic), and mass (the hospital gown is heavier). We test each participant with two different arm poses, randomly chosen from three poses. The poses and garments we use are shown in Figure 2.16. We evaluate each of the three methods for two trials on each pose-garment combination, resulting in a total of 24 trials per participant. We also randomize the ordering of the test poses, garments, and methods. Participants are asked to hold their arm steady throughout each trial. We run each trial for a fixed number of time steps, unless the participant asks to stop or the measured force is above a safety threshold (15 Newtons). We train a single force dynamics model and test it on all these 10 participants.

Force dynamics model training: Before evaluating FCVP in the human study, we need to capture real-world force data and train a force dynamics model. We capture force data from a separate 11-participant study, using a similar procedure as noted above. The 11 participants consist of 6 males and 5 females with ages ranging from 22 to 50. For each participant, we first run the Vision with Random Actions baseline for 8 trials, 2 trials for each combination of 2 arm poses and 2 garments. Using the force data collected in these 8 trials, we train a force dynamics model on this participant. We then run our method (FCVP) with the trained force dynamics model, as well as the Vision Only baseline (order randomized) for 8 trials on the same arm poses and

	Arm Dressed Ratio ↑	Average Force Violation (N) ↓
FCVP (Ours)	$\mathbf{0.81 \pm 0.21}$	**0.089**
Vision Only [107]	0.71 ± 0.17	0.39
Vision w/ Random Action	0.71 ± 0.18	0.34

Table 2.7: Quantitative results of the human study. FCVP not only achieves higher arm dressed ratio, but also has significantly lower force violations.

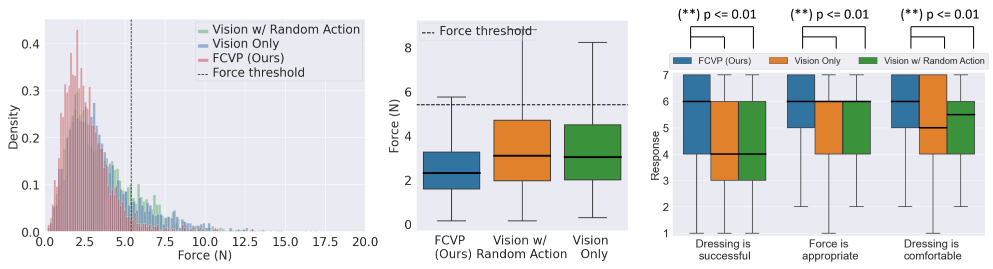


Figure 2.17: Left and middle: Density plot and box plot of the force distributions on all participants in the human study. The dashed black line represents the force threshold. Our method greatly reduces the force violation compared to the baselines. Right: Likert item responses from all 10 participants. FCVP achieves statistically significant differences from both baselines with higher reported scores for all 3 Likert items.

garments. This results in 24 dressing trajectories for each of the 11 participants. By using 3 different methods, we are able to enlarge the distribution of the captured force data, which is beneficial for training a single generalized force dynamics model. These 3 methods used for data collection are optimized for either completing the dressing task (Vision Only baseline and Vision with Random Action baseline), or reducing the force (FCVP), thus they are all safer compared to random trajectories, reducing the safety risk posed to the participants during data collection. We ask participants to hold their arm steady throughout each trial when collecting the force data to ensure the accuracy of the collected data. We use the force data captured during the 264 dressing trials on these 11 participants to train a single generalized force dynamics model, and evaluate it in another human study with 10 new participants (gender and age distributions as described above). Note that the 10 new participants we test in the evaluation of FCVP with the generalized force dynamics model are all different from the 11 participants used for collecting the force data. More details of the study procedure for both human studies can be found in our project website.

Human Study Results: Videos of the dressing trials are available on our project website. Table 2.7 compares the results of all methods, averaged over all 10 participants. As shown, our Force-Constrained Visual Policy has significantly lower force violation compared to both baselines. Furthermore, it achieves higher arm dressed ratios. Due to sim2real transfer gaps, the vision-based policy often deviates to a state where the garment gets caught on the person. This state usually does not occur in simulation, as the simulated garments are often more elastic with lower frictional coefficients than those in the real world, due to limited simulation fidelity. By constraining the amount of force applied to the person, the force dynamics model guides the vision-based policy to avoid such scenarios, and as a result, the final dressing performance is higher (see Figure 2.13 for an example). This same factor might affect other multimodal learning baseline methods as well, potentially leading to a similar discrepancy between their sim2sim and sim2real performance. Still, the performance of the other baseline methods would likely be lower than that of our method (FCVP) in the real world, since their performance in simulation is quite poor (see Fig 2.15). Further, due to the large amount of training data (thousands of dressing trajectories) needed for fine-tuning other multimodal learning methods (see the last column of the table in Fig 2.15), it is often impractical to train and test these methods in the sim2real setting. In contrast, our method is able to efficiently learn a dynamics model from just 264 real world trajectories. Thus, our method is both higher performing and more practical than the other approaches. We note that the performance of the Vision Only baseline is lower than reported by [107]. This is due to us testing on only a subset of the garments and poses (the most difficult ones) among those tested in [107].

Figure 2.17 shows the density (left) and box (middle) plots of the force distributions of all participants. As shown, most of the forces exerted by FCVP are below the force threshold, while the other two baselines violate the threshold more frequently, demonstrating the strong effectiveness of our method for reducing the force violation.

Figure 2.17 (right) also reports the Likert Item responses of all participants. On average, FCVP achieves a median score of 6 for all three Likert items, meaning that the participants "Agree" that the robot successfully dressed the garment onto their arm, "Agree" the force the robot applied to them during dressing is appropriate, and "Agree" that the dressing process was comfortable for them. The medians of FCVP are higher than or the same as both baselines for all 3 Likert items.

We conduct a Wilcoxon signed rank test to test if the distribution of the paired difference in scores between two methods is different from a distribution symmetric about zero. For all three Likert items and both baselines, we obtain a p-value smaller than 0.01 ($p < 0.01$), i.e., we find a statistically significant difference between FCVP and the two baselines, and the median of the difference is greater than zero.

We note that our human study is performed on a younger age distribution than the group that will likely need assistive dressing. However, as our experiments have shown, the dynamics model is able to generalize to new participants within the same age distribution; thus we believe that if it is trained with an older age distribution, it should perform well on that older age distribution as well. We hope to verify this in future studies.

Generalization of the force dynamics model. As the input to the force dynamics model is the partial point cloud of the scene, which captures the shape and size of the human arm, the force dynamics model should be able to generalize to the shape and size of the arm within the training distribution. In our human study, the size and shape of the arms of the 10 evaluation participants are different to those of the 11 training participants. The average prediction error (L1 norm) of the force dynamics model is 0.00631N on the 11 training participants, and 0.0478N on the 10 evaluation participants. Despite this train-eval gap, the evaluation prediction error is still small (< 0.1N), and the force dynamics model still proves to be useful in reducing force violations, as shown in Table 2.7. These results indicate that the force dynamics model can generalize reasonably well to the shape and size of the human arms. We believe the gap can be narrowed in future work by regularization, or collecting more data.

We also analyze the generalization of the force dynamics model to different physical properties of the garments, such as roughness and elasticity, in simulation. We create 50 garments with the same geometry but different elasticity, by randomly sampling the spring coefficients of the garments within a range of $[0.3, 1.5]$. We then train a force dynamics model on 40 garments and test on the remaining 10. The force dynamics model generalizes well to the garments with unseen elasticity: the average force violation increased slightly from 2.21 to 4.13 simulation units from training garments to unseen garments (a vision-based policy trained on the 10 unseen garments has a force violation of 38.8 simulation units).

More details of these experiments can be found on our project website.

System analysis. There can be cases where there is no action whose predicted force is below the safety threshold. We present the ratio of the number of timesteps where this situation occurs to the total number of timesteps. Findings from our human study indicate that this ratio is low, at 4.028%, showing that such cases are uncommon and generally don't impact dressing performance. The ratio could be further lowered by sampling more actions when solving the constrained optimization problem. The average inference time taken to solve the optimization problem in the real-world experiments is 0.065 seconds per timestep. Each dressing trial usually lasts between 40 and 60 seconds.

Limitations

One limitation is that our method assumes the person holds a static pose and that the robot has already grasped the garment. Prior works have introduced new sensors and control strategies specifically targeting at relaxing these assumptions [30, 34, 130], which could be applied in conjunction with our work. Additionally, in our experiments, we note that FCVP still has some forces that exceed the threshold. There are two reasons for this: First, there may be minor errors in the learned force dynamics model's prediction. An action can be predicted to be below the threshold, but actually applies more force when executed. This issue can be mitigated by collecting more data for training the force dynamics model to make it more accurate, or by adding a buffer to the force threshold to account for prediction inaccuracies. Second, because our method uses random shooting to solve the constrained optimization problem, the solutions may not satisfy the constraint if all sampled actions are infeasible. This would result in some forces above the threshold. This issue can be alleviated by sampling more actions until a feasible action is found, or by using alternative optimization algorithms to solve the constrained optimization problem. At last, we request all participants to wear short-sleeve T-shirts during dressing. The properties of the cloth they wear, such as friction and elasticity, could affect the forces applied to the users during the dressing process and affect the ground-truth force labels used to train the force dynamics model. We have not tested if the force dynamics model generalizes to other clothing the users wear, such as pulling a jacket over a long-sleeve shirt, which we leave as interesting future work.

2.2.7 Conclusion

In this paper, we propose a new method to leverage both vision and force modalities for robot-assisted dressing, based on which we build a system that combines these modalities to ensure task progress and low applied forces for safety. We learn a vision-based policy via reinforcement learning in simulation across diverse people, poses, and garments. We train a force dynamics model directly in the real world to achieve safety and overcome inaccuracies in simulated force sensing with deformable garments. Our system combines the vision-based policy and the force model via a constrained optimization problem to find actions that progress the dressing process without applying excessive force to the person. We evaluate our system in simulation and in a real-world human study with 10 participants and 240 trials, demonstrating that it greatly outperforms prior baselines.

Chapter 3

Automatic Reward Generation for Robotic Manipulation

3.1 Reinforcement Learning from Vision Language Foundation Model Feedback

3.1.1 Introduction

One of the key challenges of applying reinforcement learning (RL) is designing an appropriate reward function that will lead to the desired behavior. This procedure, known as reward engineering, demands considerable human effort and trial-and-error iterations, but is often required for good results [37, 55, 81, 98]. In this work, we aim to develop a fully automated system that can generate a reward function and use it to teach agents to perform a task with RL *by using only a language description of the task*, eliminating the extensive human effort required to craft reward functions manually.

Prior work has studied replacing human supervision by prompting large language models (LLMs) to write code-based reward functions [71, 108, 118]. However, these methods usually assume access to the environment code, rely on the low-level ground-truth state information for reward generation, and face challenges with scaling up to high-dimensional environments and observations, such as manipulating complex deformable objects. Others [21, 53] extract an intrinsic reward and combine it with the task reward using preference labels generated by an LLM comparing text descriptions of two agent states. However, text descriptions of the states can be non-trivial for certain tasks, such as manipulating deformable objects, as the exact states are hard to describe accurately using language. Further, these works rely on the ground-truth low-level state information to generate the text descriptions of the states, which may not be easily accessible.

Another related line of work obtains rewards from visual observations by using contrastively trained vision language models, such as CLIP [91], to align image or video observations with task descriptions in a learned latent space [2, 24, 70, 73, 92, 99]. However, the reward signals produced in these works are often of high variance and noisy [73, 99]. As a result, prior work often has to fine-tune these CLIP-style models for their specific tasks at hand [70, 73].

To this end, we present RL-VLM-F, a method that *automatically* generates reward functions for agents to learn new task. RL-VLM-F (Figure 3.1) requires only a single text description of the task goal and the agent's visual observations, leveraging vision language foundation models (VLMs) that are trained on diverse, general text and image corpora (e.g., GPT-4V [80], Gemini [103]). The key to our approach is to query these models to give *preferences over pairs of the agent's image observations* based on the text description of the task goal and then learn a reward function from the preference labels, rather than directly prompting these models to output a raw reward score, which can be noisy and inconsistent [92, 99]. This allows us to draw from the rich literature on reinforcement learning from human preferences [20, 56, 112], without requiring actual humans, to train reward functions automatically for new tasks. Furthermore, by using a VLM to compare image observations instead of text descriptions of the states, RL-VLM-F does not need access to the low-level ground-truth states for reward generation and can be applied to complex tasks involving deformable objects where accurate text description of the states are non-trivial. We test our method on 7 tasks involving classic control, rigid, articulated, and deformable object manipulation. We show that our approach can produce reward functions that lead to policies that solve diverse tasks, and our approach substantially outperforms prior methods and alternative ways to use VLMs to generate rewards. We also perform extensive analysis and ablation studies to provide insights into RL-VLM-F's learning procedure and performance gains. In summary, we make the following contributions:

- We propose RL-VLM-F, a method that *automatically* generates reward functions for agents to learn new tasks, using only *a text description of the task goal and the agent's visual observations*, eliminating the extensive human effort involved in manually crafting reward functions.
- We show that RL-VLM-F can be used to generate reward functions and learn policies that can solve a series of rigid, articulated, and deformable object manipulation tasks, and it greatly outperforms prior methods.
- We perform extensive analysis and ablation studies to provide insights into RL-VLM-F's learning procedure and performance gains.

3.1.2 Related Work

Inverse Reinforcement Learning

Similar to our work, inverse reinforcement learning (IRL) aims to learn a reward function that can be used to train a policy to solve tasks. IRL methods usually learn a reward function from expert demonstrations [1, 33, 43, 77, 78, 132]. In contrast, while RL-VLM-F also learns a reward function to train a policy, it only requires a text description of the task goal and does not require collecting expert demonstrations.

Learning from Human Feedback

Another line of work directly learns a reward function from human feedback, in the form of pairwise trajectory preference or ranking comparisons, to train a reward function [11, 12, 13, 20, 46, 56, 60, 75, 112]. In most cases, human preferences and rankings of robot trajectories are easier to collect than demonstrations of robot trajectories. However, because each comparison conveys little information on its own, many preference queries are needed before the reward function is well-trained enough to train an agent to perform the task. RL-VLM-F instead queries a VLM to perform the comparison to train a reward function, removing the need for extensive human labor in giving preference labels.

Large Pre-trained Models as Reward Functions

Kwon et al. [54] first demonstrated that large pre-trained models—large language models (LLM) specifically—can generate rewards for RL agents in text-based tasks. Other works followed by demonstrating that LLMs can write structured code for training robots [124] or directly write Python code for training many kinds of agents [71, 108, 118]. However, many tasks are challenging to write reward functions for. For example, cloth folding requires tracking the locations of many individual cloth keypoints, which can change from one folding task to another. In these instances, visual reasoning is better suited for understanding how to reward the agent. RL-VLM-F queries a VLM to compare agent observation *images* so that it can use visual observations to reason about how well the agent is progressing in a task. In addition, prior methods usually assume access to the environment source code when writing the reward functions, whereas our method does not require such assumptions.

Another line of prior works rewards agents from image observations by aligning agent trajectory images with task language descriptions or demonstrations with contrastively trained visual language models [23, 31, 70, 76, 79, 92, 99]. However, experiments from these papers directly demonstrate that contrastive alignment is noisy and its accuracy relies heavily on the input task specification and how well-aligned the agent observations are to the pre-training data [70, 76, 92, 99]. Further, CLIP-style models have thus far been limited to outputting noisy raw scores. We demonstrate that using preferences results in superior performance to outputting raw scores, shown in our experiments in Section 3.1.6. Finally, our work shares a similar idea to RLAIF [7], which proposed to mix preference labels generated by an LLM and a human in the context of fine-tuning LLMs, and Motif [53], which proposed to generate intrinsic rewards using preference feedback from an LLM in the game of NetHack based on ground-truth text descriptions of the game state. In contrast, we use a VLM to generate the preference labels without any human labeling and learn the reward function from visual image observations without the need to access ground-truth states, focus on the domain of robotics control and manipulation, and directly generate task rewards instead of intrinsic rewards.

3.1.3 Background

We consider the standard Markov Decision Process and reinforcement learning setup [102]. At every timestep t, the agent receives a state s_t from the environment and chooses an action a_t based on a policy $\pi(a_t \mid s_t)$. The environment gives a reward r_t after the agents executes action a_t and transitions to s_{t+1}. The goal of the agent is to maximize the return, which is defined as discounted sum of rewards $R = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)$ with discount factor γ.

Preference-based reinforcement learning. Our work builds upon preference-based RL, in which an agent learns a reward function from preference labels over its behaviors [20, 46, 56, 57]. Formally, a segment σ is a sequence of states $\{s_1, ..., s_H\}$, $H \geq 1$. In this paper we consider the case where the segment is represented using a single image, i.e., $H = 1$. Given a pair of segments (σ^0, σ^1), an annotator gives a feedback label y indicating which segment is preferred: $y \in \{-1, 0, 1\}$, where 0 indicates the first segment σ^1 is preferred, 1 indicates the second segment is preferred, and -1 indicates they are incomparable or equally preferable. Given a parameterized reward function r_ψ over the states, we follow the standard Bradley-Terry model [14] to compute the preference probability of a pair of segments:

$$P_\psi[\sigma^1 \succ \sigma^0] = \frac{\exp\left(\sum_{t=1}^{H} r_\psi(s_t^1)\right)}{\sum_{i \in \{0,1\}} \exp\left(\sum_{t=1}^{H} r_\psi(s_t^i)\right)}, \tag{1}$$

where $\sigma^i \succ \sigma^j$ denotes segment i is preferred to segment j. Given a dataset of preferences $D = \{(\sigma_i^0, \sigma_i^1, y_i)\}$, preference-based RL algorithms optimize the reward function r_ψ by minimizing the following loss:

$$\begin{aligned}\mathcal{L}_{\text{Reward}} = -\, \mathbb{E}_{(\sigma^0, \sigma^1, y) \sim \mathcal{D}} \Big[& \mathbb{I}\{y = (\sigma^0 \succ \sigma^1)\} \log P_\psi[\sigma^0 \succ \sigma^1] \\ & + \mathbb{I}\{y = (\sigma^1 \succ \sigma^0)\} \log P_\psi[\sigma^1 \succ \sigma^0] \Big].\end{aligned} \tag{2}$$

In preference-based RL algorithms, a policy π_θ and reward function r_ψ are updated alternatively: the reward function is updated with a dataset of preferences as described above, and the policy is updated with respect to this learned reward function using standard reinforcement learning algorithms. Specifically, we use PEBBLE [56], a preference-based RL method with unsupervised pre-training and off-policy learning, as the underlying preference-based RL algorithm.

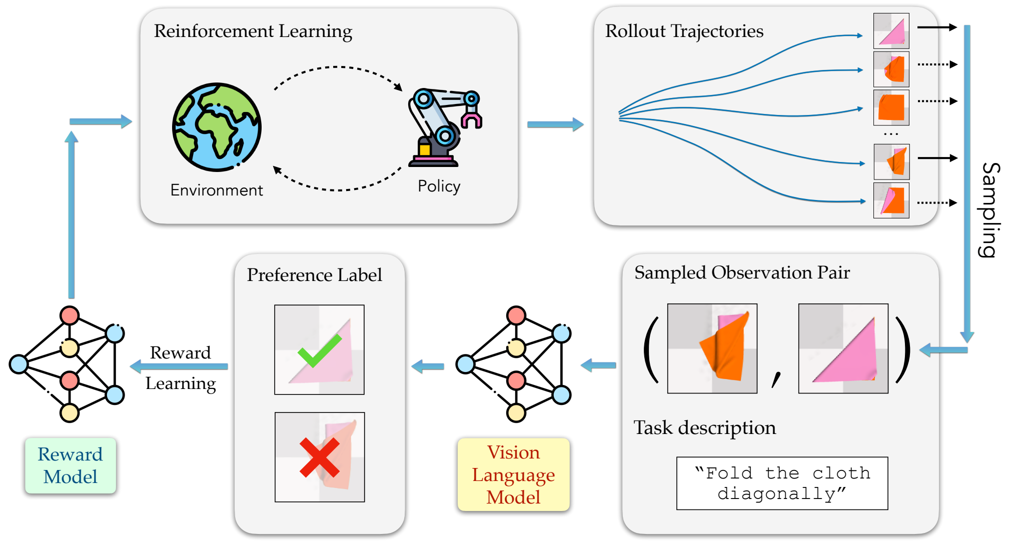


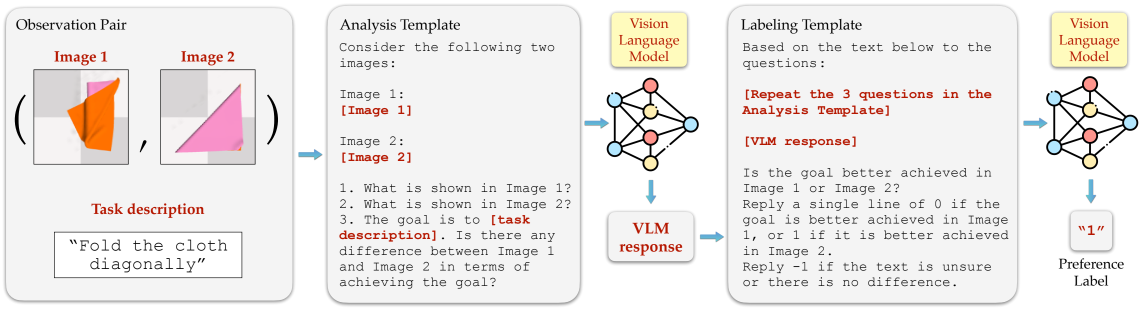
Figure 3.1: RL-VLM-F automatically generates reward functions for policy learning on new tasks, using only a text description of the task goal and the agent's visual observations. The key to RL-VLM-F is to query VLMs to give preferences over pairs of the agent's image observations based on the text description of the task goal, and then learn a reward function from the preference labels.

3.1.4 Assumptions

We make the following assumptions on the VLMs to be used in this paper: 1) We assume that the VLMs have been trained on diverse text and image corpora, enabling them to generalize well and reason across various environments and tasks. 2) The VLMs should be capable of processing multiple images simultaneously and performing comparative analyses on pairs of images as this is crucial for generating preference labels. 3) RL-VLM-F is designed to operate on tasks for which the quality or success of a state can be discerned from a single image or a sequence of images. We consider large pretrained vision-language foundation models, such as Gemini [103] and GPT-4 Vision [80], to satisfy these assumptions.

3.1.5 Method

Figure 3.1 provides an overview of RL-VLM-F. Unlike previous preference-based RL algorithms that require a human annotator to give the preference labels, RL-VLM-F leverages a VLM to do so based solely on a text description of the task's goal, thus



Analysis Template

```
Consider the following two
images:

Image 1:
[Image 1]

Image 2:
[Image 2]

1. What is shown in Image 1?
2. What is shown in Image 2?
3. The goal is to [task
description]. Is there any
difference between Image 1
and Image 2 in terms of
achieving the goal?
```

Labeling Template

```
Based on the text below to the
questions:

[Repeat the 3 questions in the
Analysis Template]

[VLM response]

Is the goal better achieved in
Image 1 or Image 2?
Reply a single line of 0 if the
goal is better achieved in Image
1, or 1 if it is better achieved
in Image 2.
Reply -1 if the text is unsure
or there is no difference.
```

Figure 3.2: We use a two-stage VLM-querying process for generating preference labels to train the reward function. In the analysis stage, we query the VLM to generate free-form responses describing and comparing how well each of the two image observations achieves the task goal. Then, in the labeling stage, we prompt the VLM with the VLM-generated text responses from the first stage to extract a preference label between the two image observations. The template shown here is the actual entire template we use for all experiments.

automating preference-based RL and mitigating the time-intensive human supervision required in writing reward functions or providing preference labels. RL-VLM-F works as follows: first, the policy π_θ and the reward function r_ψ are randomly initialized. Given a task goal description, our method then iterates through the following cycle: (1) The policy π_θ is updated using RL with the reward function r_ψ, interacts with the environment, and stores image observations into a buffer; (2) A batch of image pairs is randomly sampled from the stored buffer and sent to a VLM. The VLM is queried to produce preference labels for these image pairs in terms of which one better performs the task based on the text description of the task goal; (3) The reward model is updated with the loss in Equation (2) using the preference labels produced by the VLM. The full detailed procedure of RL-VLM-F can be found in Algorithm 1.

Prompting VLMs to generate preference labels for reward learning

To train the reward model r_ψ, we first need to generate preference labels from the VLM. To do this, we sample two images from the "image observation buffer" $\mathcal{I}$, which stores image observations of the policy during learning, and then query the VLM for which of the two images better performs the task according to the text goal description (Algorithm 1 lines 17-18). The querying process is illustrated in Figure 3.2. It consists of two stages: an **analysis** stage and then a **labeling** stage. In the **analysis** stage, we query the VLM to generate free-form responses describing

Algorithm 1 RL-VLM-F

```
input Text description of task goal $l$
 1: Initialize policy $\pi_\theta$ and reward $r_\psi$
 2: Initialize the preference buffer $\mathcal{D} \leftarrow \emptyset$, RL replay buffer $\mathcal{B} \leftarrow \emptyset$, image observation buffer $\mathcal{I} \leftarrow \emptyset$,
    policy gradient update steps $\mathcal{N}_\pi$, reward gradient update steps $\mathcal{N}_r$, VLM query frequency $K$,
    number of preference queries per time $M$
 3: for each iteration $iter$ do
 4:   // Policy learning and data collection
 5:   for $t = 1$ to $T$ do
 6:     Collect state $s_{t+1}$, image $I_{t+1}$ by taking $a_t \sim \pi_\theta(a_t|s_t)$
 7:     Add transition $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, s_{t+1}, r_\psi(s_t))\}$
 8:     Add image observation $\mathcal{I} \leftarrow \mathcal{I} \cup \{I_{t+1}\}$
 9:   end for
10:   for $n = 1$ to $\mathcal{N}_\pi$ do
11:     Sample random batch $\{(s_t, a_t, s_{t+1}, r_\psi(s_t))_j\}_{j=1}^{B} \sim \mathcal{B}$
12:     Optimize policy $\pi_\theta$ using the sampled batch with any off-policy RL algorithm
13:   end for
14:   // Preference by VLM and Reward learning
15:   if $iter \% K == 0$ then
16:     for $m = 1$ to $M$ do
17:       Randomly sample two images $(\sigma^0, \sigma^1)$ from buffer $\mathcal{I}$
18:       Query VLM with $(\sigma^0, \sigma^1)$ and task goal $l$ for label $y$
19:       Store preference $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\sigma^0, \sigma^1, y)\}$
20:     end for
21:     for $n = 1$ to $\mathcal{N}_r$ do
22:       Sample minibatch $\{(\sigma^0, \sigma^1, y)_j\}_{j=1}^{D} \sim \mathcal{D}$
23:       Optimize $r_\psi$ in Equation (2) with respect to $\psi$
24:     end for
25:     Relabel entire replay buffer $\mathcal{B}$ using updated $r_\psi$
26:   end if
27: end for
```

and comparing how well each of the two images achieves the task goal. Then, in the **labeling** stage, we prompt the VLM with the VLM-generated *text responses* from the first stage to extract a preference label between the two images.[1] Specifically, the labeling stage prompt repeats the questions in the analysis prompt, fills in the VLM's response from the analysis stage, and then asks the VLM to generate a preference label $y \in \{-1, 0, 1\}$. We specify in the prompt that 0 or 1 indicates that the first or second image is better, respectively, and -1 indicates no discernible differences. We do not use the image pairs to train the reward model if the VLM returns -1 as the preference label. Finally, as shown at line 19 of Algorithm 1, we store the

[1]We can also use an LLM in this stage as it only requires text inputs, but for simplicity, we use the same model as for the first stage of the querying process (a VLM).

preference labels produced by the VLM into the preference label buffer $\mathcal{D}$ during the training process. Standard preference-based reward learning can then be performed (as detailed in Section 3.1.3) to train the reward function with Equation 2 using the preference buffer $\mathcal{D}$. Reward learning corresponds to lines 21-24 in Algorithm 1.

To minimize prompt engineering effort, we use a unified template *across all environments* (the exact entire template is shown in Figure 3.2). Therefore, to train a policy for a new environment with RL-VLM-F, one only needs to provide the task goal description; the labels and subsequently the reward function will then be automatically trained with the above process.

Implementation Details

For policy training, we use SAC [39] as the underlying RL algorithm. As in PEBBLE [56], we relabel all the transitions stored in the SAC replay buffer once the reward function r_ψ is updated (line 25 in Algorithm 1). We set the policy gradient update step $\mathcal{N}_\pi$ to be 1. The values of all other parameters in Alg. 1 can be found in Appendix B.2.

3.1.6 Experiments

Setup

We evaluate RL-VLM-F on a set of tasks, spanning from straightforward classic control tasks to complex manipulation tasks involving rigid, articulated, and deformable objects. The tasks are as follows.

- One task from OpenAI Gym [15]:
    - *CartPole* where the goal is to balance a pole on a moving cart.
- Three rigid and articulated object manipulation tasks from MetaWorld [122] with a simulated Sawyer robot:
    - *Open Drawer*, where the robot needs to pull out a drawer;
    - *Soccer*, where the robot needs to push a soccer ball into a goal; and
    - *Sweep Into*, where the robot needs to sweep a green cube into a hole on the table.

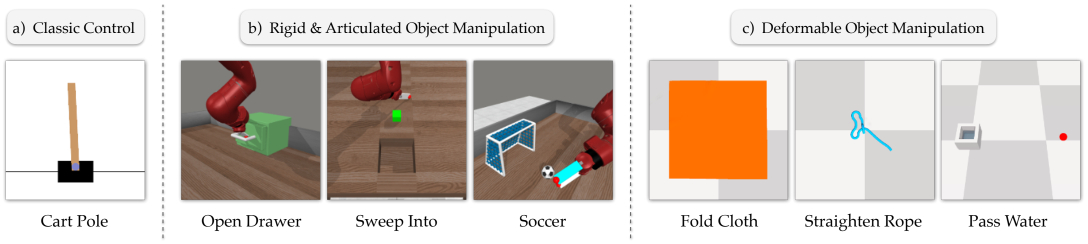


Figure 3.3: We evaluate RL-VLM-F on 7 tasks including classic control, rigid and articulated object manipulation, as well as deformable object manipulation. For *Pass Water*, the red dot represents the target location.

- Three deformable object manipulation tasks from SoftGym [65]:
  - *Fold Cloth*, where the goal is to diagonally fold a cloth from the top left corner to the bottom right corner;
  - *Straighten Rope*, where the goal is to straighten a rope from a random configuration; and
  - *Pass Water*, where the goal is to pass a glass of water to a target location without water being spilled out.

See Figure 3.3 for visualizations of these tasks. Further details about the tasks can be found in Appendix B.1.

We compare to the following baselines that make similar assumptions to us when generating the reward function, i.e., those requiring only a text description and image observations from the agents (without access to environment code). Below is a brief description of each baseline:

- **VLM Score**. Instead of querying the VLM to give preference labels over two images, this baseline directly asks the VLM to give a raw score between 0 to 1 for a given image based on the task goal description. We inform the VLM in the prompt that the score should be 1 if the task goal is perfectly achieved in the image. A reward model is then learned to regress to the scores given by the VLM.
- **CLIP Score** [92]. Given an image, the reward is computed as the cosine similarity score between the embedding of the image and the text description of the task goal using the CLIP model [91]. Such a reward computation method has also been explored in several other prior works [2, 24, 73].

- **BLIP-2 Score**. Similar to the **CLIP Score** baseline but uses BLIP-2 [63] instead of CLIP to compute the cosine similarity score.
- **RoboCLIP** [99]. This baseline uses a pre-trained video-language model, S3D [117], to compute the reward as the similarity score between the embedding of the video of the policy trajectories and a demonstration video. Since we do not assume to have access to demonstrations of the task in our method, we use the text version of RoboCLIP for a fair comparison. RoboCLIP-Text uses the pre-trained video-language model to generate rewards as the similarity score between the video embedding of the trajectory and the text embedding of the task description.
- **GT Preference**. We use the original ground-truth reward function (provided by the authors of each benchmark) to give the preference label. This should in theory serve as an oracle and upper bound on the learning performance.

Further details on the baselines, including all the text prompts we use, can be found in Appendices B.3 and B.4.

For MetaWorld tasks, we use the author-defined task success rate of the policy as the evaluation metric [122]. For all other tasks, we report the episode return of the learned policy. For all methods, the policy is learned with state observations, and we use the same policy learning hyper-parameters for all methods, i.e., the only difference between all compared methods is the reward function. For methods where a reward function needs to be learned (RL-VLM-F and VLM Score), the reward function is learned using image observations. For RL-VLM-F and the VLM Score baseline, we use Gemini-Pro [103] as the VLM for all tasks except *Fold Cloth*. We find Gemini-Pro to perform poorly on *Fold Cloth*, so we instead use GPT-4V [80] as the VLM for this task for these two methods (see Appendix B.5.2 for a comparison of Gemini-Pro and GPT-4V on this specific task). We did not run GPT-4V on all tasks due to its quota limitations. For all methods except RoboCLIP, we follow the prior work [6, 10] to remove the robot from the image for the MetaWorld tasks, as these tasks are all object-centric and removing the robot allows the VLM to focus on the target object when analyzing the images. Since these tasks are simulated, we conveniently use the simulator to make the robot transparent when rendering the images. For real-world applications, techniques such as inpainting can be used

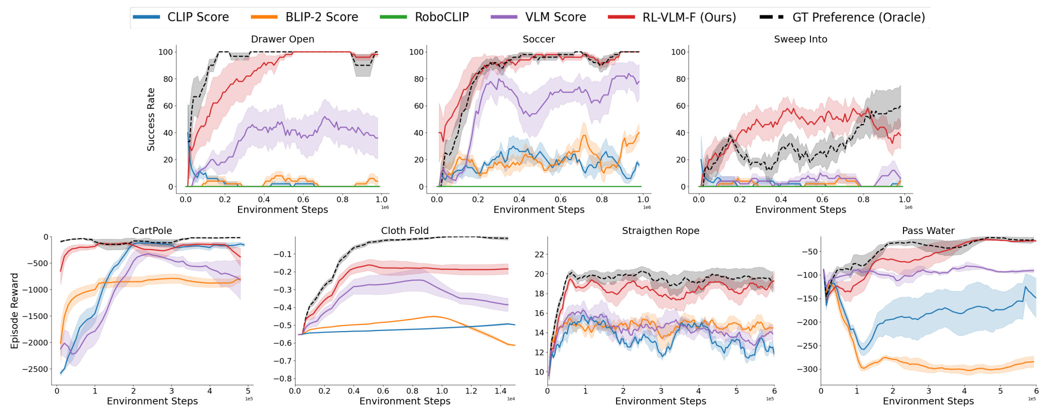


Figure 3.4: Learning curves of all compared methods on 7 tasks. RL-VLM-F outperforms all baselines in all tasks, and matches or surpasses the performance of GT preference on 6 of the 7 tasks. Results are averaged over 5 seeds, and shaded regions represent standard error. RoboCLIP is only evaluated on the MetaWorld tasks, as this is the set of tasks where the original method is evaluated.

to remove the robot from image observations. We keep the robot within the image for RoboCLIP following the original paper's setup. We test RoboCLIP only on the MetaWorld tasks, as this is the set of tasks where the original method is evaluated.

Does RL-VLM-F learn effective rewards and policies?

We first examine if RL-VLM-F leads to useful rewards and policies that can solve the tasks. The learning curves of all compared methods on all tasks are shown in Figure 3.4. As shown, RL-VLM-F outperforms all other baselines in all tasks. We find that prior approaches using CLIP or BLIP-2 score can only solve the easiest task – *CartPole*, and struggle for more complex environments, such as the rigid object manipulation tasks in MetaWorld and the deformable object manipulation tasks in SoftGym. The text version of RoboCLIP performs poorly on all three MetaWorld tasks, aligning with the original paper's results, as RoboCLIP works the best with video demonstrations available. RL-VLM-F also outperforms VLM Score in all tasks, which indicates that prompting VLMs to output a preference label for reward learning results in better task performance in contrast to treating the VLM as a reward function that outputs raw reward scores. We also observe that RL-VLM-F is able to match the performance of using GT preference in all tasks except Cloth Fold, which suggests we can use a single text description with RL-VLM-F to mitigate human

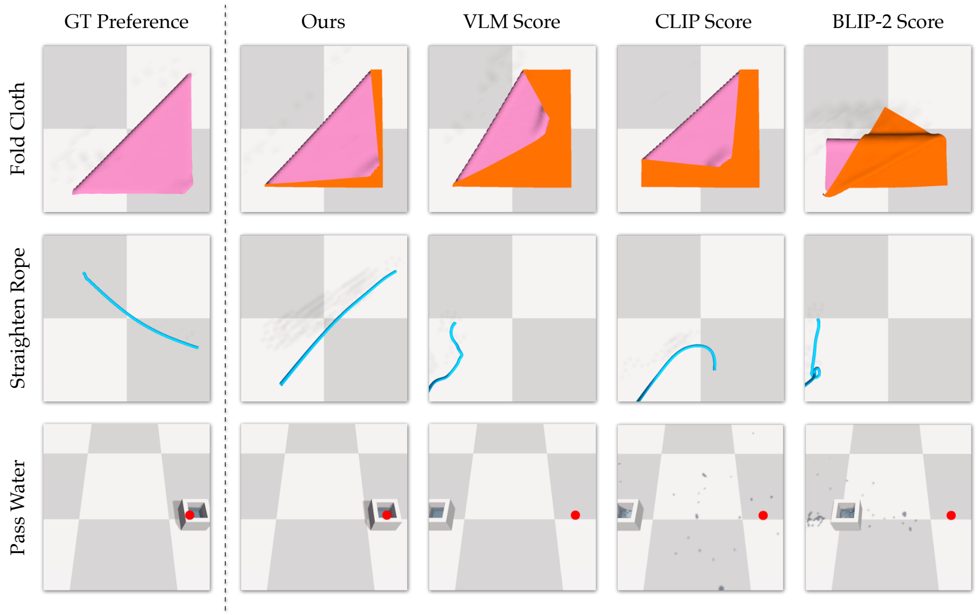


Figure 3.5: Comparison of the achieved final state of different methods on SoftGym deformable object manipuation tasks: *Fold Cloth* (Top), *Straighten Rope* (Middle), and *Pass Water* (Bottom). RL-VLM-F achieves better final states compared to all the baselines.

efforts in writing complex reward functions for these tasks.

Interestingly, for the task of *Sweep Into*, the performance of RL-VLM-F actually surpasses that of using GT preference. We suspect the reason could be as follows: the ground-truth reward function written by the authors for this task includes terms that are not directly correlated to task success. This includes a reward term for grasping the cube, which is not critical for pushing the cube into the hole.

In contrary, RL-VLM-F simply uses a text description of the task goal as "minimize the distance between the cube and the hole", thus the learned reward is less prone to bias in human-written reward functions and may better reflect the true task goal, leading to better performance.

We show the final states achieved by the policies learned with different methods on the three SoftGym deformable object manipulation tasks in Figure 3.5. As shown, for all three tasks, RL-VLM-F achieves a final state that is quantifiably better than the baselines. For *Fold Cloth*, RL-VLM-F is closest to a diagonal fold. For *Straighten Rope*, RL-VLM-F is able to fully straighten the rope and match the performance of GT preference, where all other baselines failed to fully straighten it. For *Pass Water*,

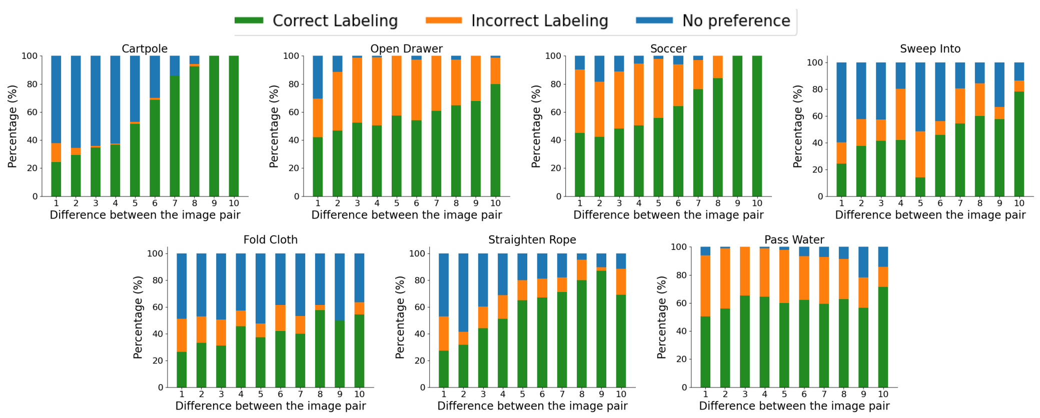


Figure 3.6: We provide analysis of the accuracy of the VLM preference labels, compared to ground-truth preference labels defined according to the environment's reward function. The x-axis represents different levels of differences between the image pairs, discretized into 10 bins, where the difference is measured as the difference between the ground-truth task progress associated with the image pairs. The y-axis shows the percentage where the VLM preference labels are correct, incorrect, or when it does not have a preference over the image pairs.

RL-VLM-F is able to transport the water to the target location without any water being spilled, and the baselines either do not move the glass, or move it in a way that spills large amounts of water.

What is the accuracy of VLM preference labeling?

Given that RL-VLM-F can learn effective rewards and policies that solve the tasks, we perform further analysis on the accuracy of the preference labels generated by a VLM. To compute accuracy, the VLM outputs $\{-1, 0, 1\}$ (no preference, first image preferred, second image preferred) which we compare to a ground truth preference label defined according to the environment's reward function. Note that we discard the image pairs with a label -1 (no preference) when training the reward model.

Our intuition is that, like humans, it would be hard for the VLM to give correct preference labels when comparing two similar images, and easier to produce correct preference labels when the two images are noticeably dissimilar in terms of achieving the goal. Figure 3.6 presents the accuracy of the VLM at various levels of differences between the two images. The "difference" between two images is measured as the difference between the ground-truth task progress associated with the images. We

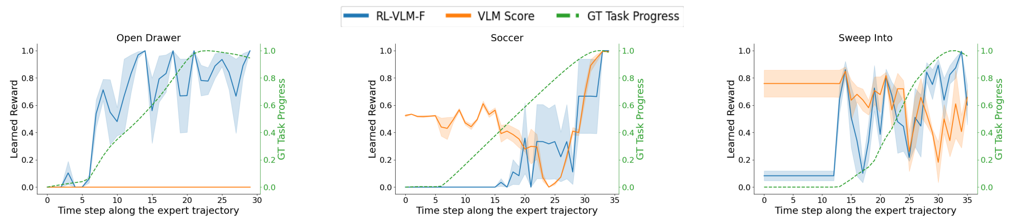


Figure 3.7: We compare how well the learned reward by RL-VLM-F and VLM Score align with the ground-truth task progress on 3 MetaWorld tasks along an expert trajectory. As shown, RL-VLM-F generates rewards that align better with the ground-truth task progress. The learned rewards are averaged over 3 trained reward models with different seeds, and the shaded region represents the standard error.

discretize the differences into 10 bins along the x axis in Figure 3.6, where a larger number indicates a greater difference between two images in terms of task progress. On the y axis, the green, orange, and blue bars represent the percentage where the VLM preference label is correct, incorrect, or when there is no preference. For all tasks, we observe a general trend of increasing accuracy, decreasing uncertainty, and decreasing error as the differences between the images increase, which aligns with intuition. This trend is most clear and consistent for the *CartPole*, *Open Drawer* and *Soccer* tasks. Overall, for all tasks, we find that the VLM is able to generate more correct preference labels than incorrect ones, and as shown in Figure 3.4, the accuracy of VLM-generated preference labels is sufficient for learning a good reward function and policy.

How does the learned reward align with the task progress?

Figure 3.7 plots the learned rewards (averaged over 3 trained reward models with different random seeds) as well as the true task progress on three MetaWorld environments along an expert trajectory that fully solves the task. Note the ground-truth task progress is not the same as the author-provided reward function: the author provided reward is a shaped version of the task progress. For *Open Drawer*, the task progress is measured as the distance the drawer has been pulled out; For *Soccer*, it is measured as the negative distance between the soccer ball and the goal; For *Sweep Into*, it is measured as the negative distance between the cube and the hole. We normalize both the ground-truth task progress and the learned reward into the range of [0, 1] for a better comparison between them. An ideal learned reward should

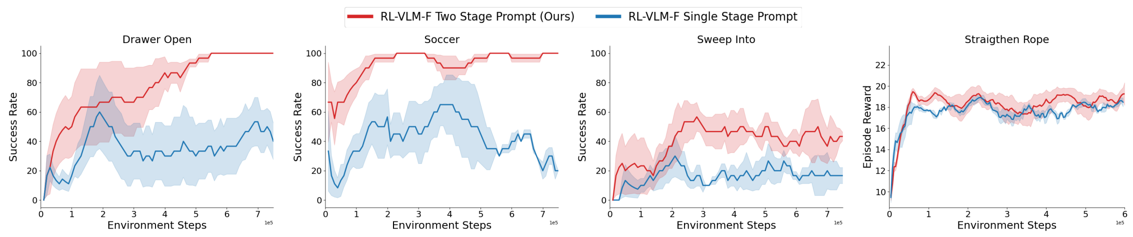


Figure 3.8: We compare RL-VLM-F with the proposed two-stage prompting strategy, and an ablated version of using a single-stage prompting strategy. The performance of the single-stage prompting is lower on 3 of the 4 tasks.

increase as the time step increases along the expert trajectory, as like the ground-truth task progress. As shown, the reward learned by RL-VLM-F aligns better with the ground-truth task progress compared with the VLM Score baseline. We do observe that the learned reward tends to be noisy and includes many local minima. Despite this, the learned reward still achieves the highest value when the task progresses the most. As shown in Figure 3.4, the learned reward is sufficient for learning successful policies. For *Open drawer*, we notice that the reward produced by VLM Score remains zero. This is likely because, during training, most of the scores given by the VLM are 0, and the model learns to predict 0 at all time steps to minimize the regression loss. We find the CLIP and BLIP-2 scores on these environments are generally noisy; the corresponding plots can be found in Appendix B.5.3.

Ablation on the prompt strategy

We used a two-stage prompting strategy for RL-VLM-F, where the VLM is first asked to analyze the pair of images in the analysis stage, and then output the preference label in the labeling stage. Here we compare it with a single-stage prompting strategy where we query the VLM to directly output a preference label over the two image observations in a single stage. The detailed single-stage prompt can be found in Appendix B.4.4. Figure 3.8 presents the comparison on 4 tasks: Open Drawer, Soccer, Sweep Into and Straighten Rope. As shown, the success rate of using the VLM with the single-stage prompt is lower than using the two-stage prompt on 3 out of the 4 tasks.

3.1.7 Conclusion and Future Work

In this work, we present RL-VLM-F, a method that automatically generates reward functions via querying VLMs with preferences given a task descriptions and image observations for a wide range of tasks. We demonstrate our proposed method's effectiveness on rigid, articulated, and deformable object manipulation tasks.

Future work could extend RL-VLM-F to an active learning context, exploring both easy and informative VLM queries for more efficient reward learning. The adaptable nature of our method allows for the integration of more advanced VLMs when they become available, potentially addressing more complex tasks. It would also be interesting to test RL-VLM-F on tasks with longer horizon. One could first decompose the tasks into subtasks with shorter horizon, either via manual decomposition or foundation models [3]. Then, RL-VLM-F can be used to solve each subtask. Additionally, our approach offers a practical pathway to applying RL in real-world settings, where obtaining reward functions is often difficult.

Chapter 4

Discussion and Future Work

In this thesis, we explore robotic manipulation of deformable objects. In the first part, we develop a robotic system for the task of assistive dressing. We employ reinforcement learning (RL) from partial point clouds, policy distillation, and guided domain randomization to enable the policy to generalize to a wide variety of body shapes, garment types, and arm poses in the real world. We further enhance the safety and comfort of our system by leveraging force sensing. We combine a vision-based RL policy trained in simulation with a force dynamics model trained with real robot data to infer actions that are both safe and effective. In the second part, we propose a novel framework that automatically generates reward functions by leveraging feedback from vision language models (VLMs) and demonstrate our method's effectiveness on deformable object manipulation.

Manipulating deformable objects has important applications in various domestic and assistive tasks. For example, in healthcare, robots can assist with dressing and undressing patients and changing bed sheets. In the home, they can help with tasks such as meal preparation and folding laundry. In addition, robots can aid in agricultural tasks such as harvesting delicate fruits and vegetables, handling plants, and sorting produce. These applications compel us to tackle manipulation problems with fewer assumptions, leading to the development of more general and robust manipulation systems. In our work on assistive dressing, we develop a vision-based dressing policy and a force-based dynamics model directly from raw sensory input, avoiding the need for low-dimensional state representations. Reinforcement learning from point cloud observations offers a versatile solution for deformable object manipulation. Additionally, in RL-VLM-F, we generate the reward function directly from high-dimensional RGB image observations, rather than relying on environment code and low-dimensional state representations as in prior work. This approach enables RL-VLM-F to effectively address deformable manipulation tasks. Furthermore, we adopt a multimodal approach in this thesis. In Chapter 2, we demonstrate that combining force and vision sensing enhances the task of assistive dressing. Visual information provides global perception, while force information helps with local control. In Chapter 3, we utilize the common sense reasoning abilities of vision language models (VLMs) by querying them with text descriptions and image observations, enabling effective reward generation.

Below, I list a few directions that are closely related to the contributions of this thesis:

- **Closed-Loop Visuo-Tactile Policy Learning:** In this thesis, we explore reinforcement learning and planning with vision and force modalities for deformable object manipulation. Another modality that offers fine-grained local information, which might not be present in visual image data due to occlusions, is tactile sensing. Therefore, an interesting future direction is multi-modal perception using vision and touch. A promising paradigm is to develop a hierarchical framework that uses vision input for a high-level policy and tactile input for a low-level reactive policy.
- **Bridging the Sim2Real Gap for Deformable Object Manipulation:** Two commonly used methods for Sim2Real are system identification [61] and domain randomization over vision or dynamics [85, 104]. System identification can be challenging due to the near-infinite degrees of freedom of deformable objects, while domain randomization can hinder policy learning and requires careful hand-tuning of the randomization range. Alternatively, we can take a task-driven domain adaptation or target-domain policy adaptation approach. For task-driven domain adaptation, we can learn an adaptation policy via meta-learning style methods in simulation to adjust the simulation parameters based on the current policy's performance. For target-domain policy adaptation, one approach is to adaptively update the policy in target-domain according

based on uncertainty estimation.

Appendix A

One Policy to Dress Them All

A.1 System Implementation Details

A.1.1 Full Reward Function

As mentioned in the main paper Section IV.B, in addition to the main reward r_m that measures the progression of the dressing task, we have three additional reward terms, detailed as follows:

- The second reward term is a force penalty r_f that prevents the robot from applying too much force through the garment to the person. Too much force will not only hurt the person in the real world but also lead to unrealistic simulation as the cloth will penetrate through the arm. Specifically, let f be the total applied force from the garment to the human, and f_{max} be the max threshold, this reward is computed as $r_f = -0.001 \cdot \max(f - f_{max}, 0)$. We set f_{max} to be 1000 units as measured by the simulator, which is an empirical threshold value when the cloth starts penetrating. We note that due to simulation modeling inaccuracies, this force number given by the simulator does not correspond to 1000 Newtons of force in the real world.
- The third reward term is a contact penalty r_c that prevents the robot end-effector from moving too close to the person. Let d_e be the shortest distance between the end-effector and the arm, this reward is computed as $r_c = -0.01 \cdot \mathbf{1}(d_e < d_{min})$,

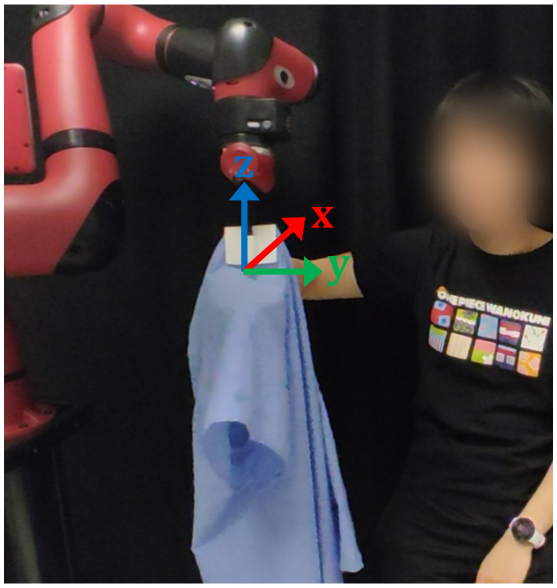


Figure A.1: Coordinate system in the real world.

where $\mathbf{1}(d_e < d_{min}) = 1$ if $d_e < d_{min}$ and 0 otherwise. d_{min} is set to be 1 cm in our experiments. This reward term, as well as the previous one, are used to keep the dressing process safe and comfortable for the person.

- The last reward term is a deviation penalty r_d that discourages the garment center from moving too far away from the arm. Let d_g be the shortest distance from the garment center p^g_{center} to the arm, this reward is given by $r_d = 0.02$ if $d_g < 3$ cm, $r_d = -0.05$ if $d_g > 7.5$ cm, and $r_d = 0$ otherwise.

The full reward function we use is: $r = r_m + r_f + r_c + r_d$.

A.1.2 Observation Domain Randomization

We pre-process the point cloud by filtering it with a voxel grid filter: we overlay a 3d voxel grid over the partial point cloud and then take the centroid of the points inside each voxel to obtain a voxelized point cloud. This preprocessing step is done both in simulation training and in the real world, which makes our method agnostic to the density of the observed point cloud and thus more robust during sim2real transfer. We use a voxel size of 6.25 cm.

As mentioned in main paper Section IV.D, we perform randomization on the garment point cloud observation for robust sim2real transfer. The randomizations we perform in simulation are as follows. Note that all these randomizations are just for the observations; the underlying simulator state or physics (e.g., the garment mesh) is not randomized.

- Random cropping of garment point cloud. To make the garment point cloud observation more aligned between the simulation and the real world, we perform cropping to remove the bottom part of the garment. After the bottom part are cropped, the remaining part (which mostly contains the sleeve part of the garment) is much more similar between simulation and the real world, reducing the sim2real gap. Formally, given the garment point cloud P^g and the human finger point p^h_{finger}, we remove garment points that are δ_1 cm lower than the human finger point, i.e., the following points are kept:

  $$\{p_i \in P^g \mid p_i[z] > p^h_{finger}[z] - \delta_1\}, \tag{A.1}$$

  where $p[z]$ is the z coordinate of the point, and z is the gravity axis (Figure A.1 shows the coordinate system we use in the real world). δ_1 is uniformly randomly sampled from $[10, 25]$ cm in our experiments.

  In the real world, we obtain the garment point cloud P^g via color thresholding. Color thresholding will usually return some additional noise points that are not part of the garment, especially when environment conditions such as lighting changes. To account for such error in the real world, we add more cropping to remove the noise points returned by color thresholding. Specifically, all garment points that are δ_2 cm above the robot gripper point P^r, or δ_3 cm forward the gripper point (the forward direction is along the x axis shown in Figure A.1) are removed. Formally, only points satisfying the following conditions are kept:

  $$\{p_i \in P^g \mid p_i[z] < P^r[z] + \delta_2,\ p_i[x] < P^r[x] + \delta_3\}, \tag{A.2}$$

  where x is the coordinate axis aligning with the forward moving direction of the robot (See Figure A.1). In our experiments, we uniformly randomly sample δ_2 from $[1, 5]$ cm, and δ_3 from $[1, 5]$ cm. In summary, we perform the cropping to only keep the following garment points:

$$\{p_i \in P^g \mid p_i[z] > p^h_{finger}[z] - \delta_1,\ p_i[z] < P^r[z] + \delta_2,\ p_i[x] < P^r[x] + \delta_3\} \quad \text{(A.3)}$$

- Random dropping of garment points. In the real world, the part of garment points that are grasped by the tool is occluded by the tool (See Fig. 4 in the main paper for an illustration of the 3D-printed grasping tool we use in the real world). To account for this issue, we remove garment points that are near the gripper point in simulation as well. Specifically, any garment points that are δ_4 cm within the gripper point P^r are removed from the observation. Mathematically, points satisfying the following condition are removed:

  $$\{p_i \in P^g \mid ||p_i - P^r||_2 < \delta_4\} \quad \text{(A.4)}$$

  In each training episode of SAC, we perform this operation of removing points near the gripper with a probability of 0.5. We set $\delta_4 = 9.375$ cm in our experiment.

- Random erosion of the garment. To account for the geometric differences between the garment in simulation and in the real world, we add random erosion on the garment depth image. The erosion is done on the depth image of the garment before de-projecting it to a point cloud. The kernel size for the erosion is randomly sampled from $[0, 3, 5, 7, 9, 11, 13, 15, 17, 19]$, where 0 means no erosion will be used.

- Random dilation of the garment. To account for the geometric differences between the garment in simulation and in the real world, we add random dilation on the garment depth image. The dilation is done on the depth image of the garment before de-projecting it to a point cloud. The kernel size for the dilation is randomly sampled from $[0, 3, 5, 7, 9, 11, 13, 15, 17, 19]$, where 0 means no erosion will be used. If the sampled erosion and dilation kernel sizes are both non-zero, we do either erosion or dilation with equal probability, and do not perform them at the same time.

- Random noise added to gripper position. In the real world, we use a side-view camera and perform camera-to-robot calibration, so we can transform garment and arm point clouds into the robot frame. On the other hand, the robot gripper position is obtained via forward kinematics and is already in the robot frame, thus does not need the calibration and is perfectly accurate in the robot frame. Due to camera calibration errors, there will be minor errors when transforming the garment and arm point clouds into the robot frame. To account for such error, we add random noise to the gripper position in simulation. As we use PointNet++ as our policy architecture and it is translation-invariant (i.e., translating the whole input point cloud does not change the output), adding a random noise to the gripper position is equivalent to adding a random noise to all the garment and arm point clouds, thus mimicking the real-world calibration error. Note that the noise is only added to the gripper point in the policy observation, but not the actual gripper in simulation. At the beginning of each SAC training episode, we uniformly sample a noise from $[-3.125, 3.125]$ cm, and the noise is added to the gripper position throughout this episode.

When deploying the policy in the real world, we perform the cropping of the garment point cloud and fix $\delta_1 = 15$ cm, $\delta_2 = 2$ cm, and $\delta_3 = 1$ cm. We do not perform random dropping, erosion, or dilation of the garment points in the real world, and we do not add noise to the gripper position in the real world.

A.1.3 Simulation Training Details

We use PointNet++ [88] as both the policy and the Q function network architecture. We use implementation of PointNet++ in Pytorch Geometric [32]. For the dense transformation policy, we use the segmentation-type PointNet++. We use 2 set abstraction layers followed by a global max pooling layer, 3 feature propagation layers, and followed by a Multi-layer Perceptron (MLP). The set abstraction radius are 0.05 and 0.1, and the sampling ratios are 1 and 1. The numbers of nearest neighbors for feature propagation layers are 1, 3, and 3. The final MLP is of size $[128, 128]$. For the Q function, we use the classification-type PointNet++. We use 2 set abstraction layers, followed by a global max pooling layer, and a MLP. The set abstraction radius are 0.05, 0.1, and the sampling ratios are 1, 1. The final MLP is of size $[128, 128]$.

We use SAC [39] as the RL training algorithm. The learning rates for the actor, critic, and the entropy temperature alpha are all $1e-4$. The delayed actor update

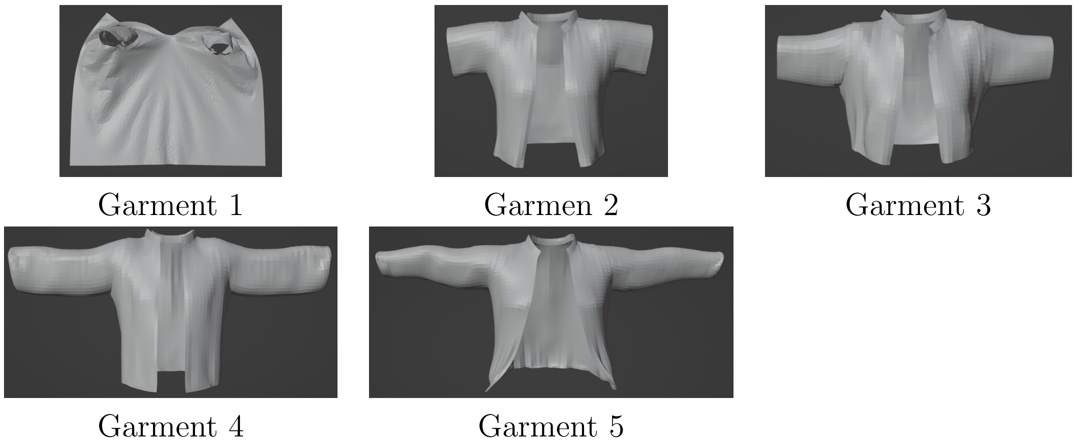


Figure A.2: Garments we used in simulation training. The first one is a hospital gown, followed by 4 cardigans. The garments are in ascending order in terms of sleeve length.

frequency is 4, i.e., we update the actor once every time the critic is updated 4 times. We use a replay buffer size of 400000, and a batch size of 64. The soft update parameter τ for the critic target network is set to be 0.01. Each episode in simulation has 150 time steps. We use the Adam [51] optimizer to train both the policy and the Q network.

A.2 Simulation Experiments

A.2.1 Experimental Setup

Human Mesh Generation

We use SMPL-X [84] model to generate human meshes of different body sizes and shapes. The body shape of the mesh is controlled by a 10D latent vector β. We uniformly sampled β from $[-2, 5]$. The height of the mesh is uniformly sampled from $[1.5, 1.9]$. The gender is uniformly sampled from [Male, Female]. Some example generated meshes are shown in Figure A.3.

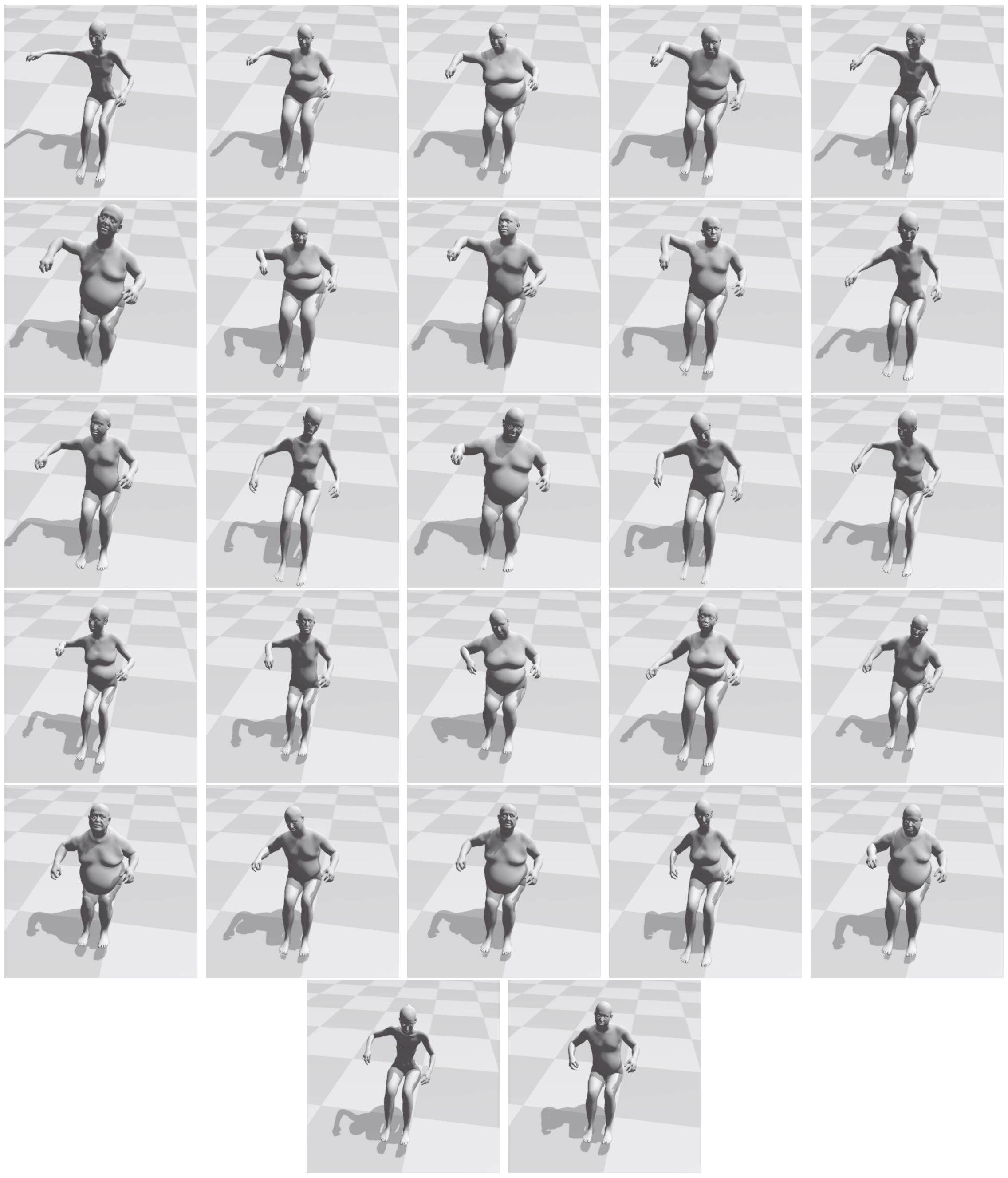


Figure A.3: Some example human meshes we generate for simulation training. From top left to bottom right: a mesh from arm pose sub-range 1 to a mesh from arm pose sub-range 27.

Simulation Garments

We use a hospital gown and 4 cardigans chosen from the Cloth3D [9] dataset. These garments have different geometries and are shown in Figure A.2. We scale the garment mesh so they are of proper size for dressing.

Arm Pose Range Decomposition

For the shoulder joint ϕ_1, we decompose the full interval $[-20, 30]$ to 3 intervals $[-20, -8], [-8, 18], [18, 30]$. For the inwards-outwards elbow joint angle ϕ_2, we decompose the full interval $[-20, 20]$ to 3 intervals $[-20, -8], [-8, 8], [8, 20]$. For the upwards-downwards joint angle ϕ_3, we decompose the full interval $[-20, 30]$ to 3 intervals $[-20, -3], [-3, 14], [14, 30]$. The cross product of all of these intervals results in 27 pose sub-ranges. Figure A.3 shows an example mesh from each of the pose sub-ranges.

Simulator Parameters

We use SoftGym [65] based on the NVIDIA FleX simulator. FleX simulates cloth as a collection of particles connected by springs. We set the stiffness of the stretch, bend, and shear spring connections to 0.3, 0.3, 0.3, respectively. The particle radius is 0.625 cm. The particle, dynamic, and static fricition coefficient are set to be 0.3, 0.3, and 0.3, respectively.

A.2.2 Baseline Implementation

Direct Vector

For the policy, we use a classification-type PointNet++. We use 2 set abstraction layers, followed by a global max pooling layer, and a MLP. The set abstraction radius are [0.05, 0.1], and the sampling ratios are [1, 1]. The final MLP is of size $[256, 256, 256, 128, 128, 128, 128, 128, 128]$. Since the Direct Vector policy does not have the feature propogation layers, the final MLP is set to be larger so the total number of parameters for the Dense Transformation and the Direct Vector policy is similar. We train Direct Vector policy using the same SAC parameters as training the Dense Transformation Policy.

TD-MPC

We use the official implementation from the authors[1]. For the MPPI planning, we use 8 iterations with 512 samples per iteration. The number of elites is 64, and planning horizon is 5. We use a classification-type PointNet++ (the same as the one used for Direct Vector policy) to encode the partial point cloud into a latent vector. The dimension of the latent vector is 100. Other hyper-parameters are set to the default values as in TD-MPC.

PCGrad

We use a re-implementation of PCGrad [121] in PyTorch [2]. The dressing task can be viewed as multi-task learning where each arm pose sub-range is treated as a separate task. As we have 27 arm pose sub-ranges, we keep 27 replay buffers, one for each sub-range. We also follow [121] to learn an individual entropy temperature α in SAC for each task. We wrap the Adam [51] optimizer with PCGrad. In each training step, we randomly sample 16 replay buffers. For each of the sampled replay buffer, we randomly sample 4 transition tuples, $\{o_n, a_n, r_n, o'_n\}_{n=1}^4$, to form a stratified batch of size 64, and use PCGrad to perform the gradient update.

Deep Haptic MPC

We re-implemented Deep Haptic MPC [28] for our environment. We train an one-step force prediction model $F(x_{1:t}, a_{t+1})$ given the robot end-effector state history $x_{1:t}$ and action a_{t+1} following the original paper. The end-effector state measurements $x_t = (\rho, v, f) \in \mathbb{R}^{13}$ at time t include the 6D position ρ, 6D velocity v, and force f applied at the robot's end-effector. We do not use any visual information to train the force prediction model. During test time, we define a cost function that encourages lower force applied to the human arm and higher dressing performance. Since the original paper assumes a single fixed arm pose, sampling actions whose velocity lies within a hemisphere facing the $+x$ global coordinate axis will encourage task progression. Different from [28], we have a diverse set of arm poses. Therefore, we use the position of the human hand, elbow, and shoulder to guide the action sampling.

[1]https://github.com/nicklashansen/tdmpc
[2]https://github.com/WeiChengTseng/Pytorch-PCGrad

At each time step, if the garment is on the forearm, we sample $N = 128$ candidate actions whose dot product with the direction from the human's finger to the elbow is positive; if the garment is on the upper arm, we sample $N = 128$ candidate actions whose dot product with the direction from the human's elbow to the shoulder is positive. We always sample candidate actions that point to the direction of task progression. The input to the cost function includes a history of robot end-effector's measurements $x_{1:t}$ and the next candidate action sample a_{t+1}. The cost function penalizes actions that lead to large forces and actions whose magnitude is large. It also encourages actions that move forward along the human arm. Formally, the cost function is represented by three weighted terms as follows:

$$\begin{aligned} J(x_{1:t}, a_{t+1}) = w_1 \; & |F(x_{1:t}, a_{t+1}|_1| \\ & - w_2 \; \bar{d} \cdot a_{t+1}^{\text{translation}} \\ & + w_3 \; ||a_{t+1}||_2^2 \end{aligned}$$

, where d is the vector from human's finger to elbow when the garment is still on forearm and from human's elbow to shoulder when the garment is on the upper arm. We run this baseline on three selected pose regions. For each region, we collect training trajectories on the first 45 poses and 5 garments to train the force prediction model and we report the evaluation results on the last 5 poses in Table I in the paper. This baseline lacks the ability to generalize to diverse poses since it does not take in as input the visual observation of the arm and the garment. We find it perform worse than our proposed method on all three selected pose regions.

Heuristic Motion Planning

As mentioned in the main paper, the motion planning baseline finds a collision-free robot end-effector path along the human arm. The following constraints are used for planning the path. First, the normal of the grasping tool (mounted on the robot end-effector) should align with the normal of the arm plane, which is the plane defined by the person's finger, elbow, and shoulder points. Second, the forward direction of the end-effector should align with the direction of the forearm (the line connecting finger and elbow points) or the direction of the upper arm (the line connecting the elbow and shoulder points). Third, the path should go through three way points, which are defined as points above a certain distance from the finger, elbow, and shoulder points. Forth, the path needs to have no collision with the human arm. This baseline's average upper arm dressed ratio is 0.32 on all 27 arm pose regions, compared to 0.68 with our method (Table II in the paper). We find such a method to perform well when there is no sharp bending around the shoulder and the elbow, and worse when the bends are sharp, aligning with findings in prior work [50].

A.3 Real-world Experiments

A.3.1 Human Study Procedure

Human arm capture. To capture only the human arm point cloud, we manually select 3 pixels on the depth image that correspond to the shoulder, elbow, and hand point of the participant (this step can be replaced by using a human pose estimator in future work). We then transform these three pixels to three points in the robot frame. We form two lines: the first connects the hand point and the elbow point (forming a line along the forearm), and the second connect the elbow point and the shoulder point (forming a line along the upper arm). Then, we crop the point cloud by only keeping points whose distances to either of these two lines are smaller than a threshold Δ. We set $\Delta = 7.5$ cm in our experiment.

Dressing time cost. Each dressing trial lasts between 1 to 2 minutes. It takes roughly 1 minute (as mentioned above) to perform the arm cropping and moving the Sawyer to be near the participant's hand. The actual dressing time is roughly between 40 seconds and 80 seconds. It usually takes 1 hour to 1.5 hours to finish the 30 dressing trials for each participant (including all other time cost such as rest time for the participant).

Scripts for participant. Before the study begins, we read and show the following script to the participant for him/her to get familiar with the study procedure:

We are conducting a study to evaluate a robot-dressing system. The robot will dress the garment on your right arm. We will now walk you through the steps we are taking. You will first read and sign the consent form, and fill a demographic form. We will then measure some statistics of your arm, including the forearm length, upper

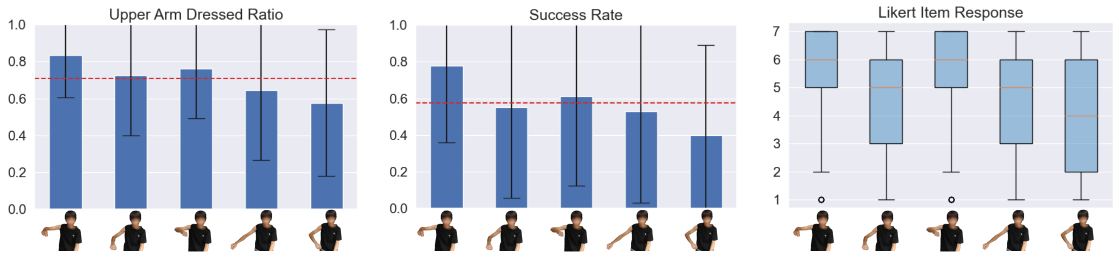


Figure A.4: Upper arm dressed ratio, success rate, and Likert item responses for different poses in the human study. The results are averaged over all 425 dressing trials from 17 participants in the human study.

arm length, and the arm circumference before the study starts. We will put a marker on your shoulder for the experiment. We will then start the study. There will be 30 dressing trials, 6 trials for 5 garments. Each trial will be on a different arm pose, and a different garment. You will be asked to hold your arm static during the trial. We will show the arm pose you need to hold on a computer screen, and we will also demonstrate the pose by ourselves to you if it's not clear. After we tell you to start, you need to hold the pose static for 1 - 2 minutes while we operate the robot to perform the dressing. Very occasionally the robot gripper might have contact with you during the dressing process. If you feel uncomfortable during the trial, you can tell us to stop the trial any time. Occasionally there will be operation failures on us for a trial. We will repeat those trials when such failures happen. After each trial, please keep holding the arm while we measure some statistics of the dressing performance. Then you can rest the arm and you will be asked to fill a questionnaire. The questionnaire is a Likert item stating that "The robot successfully dressed the garment onto my arm". 1 is strongly disagree and 7 is strongly agree. You can rest whenever you feel tired, just let us know. After 6 trials for each garment, we will change the garment and you can also rest.

A.3.2 More Real-world Experimental Results

We present the upper arm dressed ratio, success rate, and Likert item responses for each of the poses we tested in the human study in Figure A.4. Again, we notice that some poses (the 5th one) is harder than others.

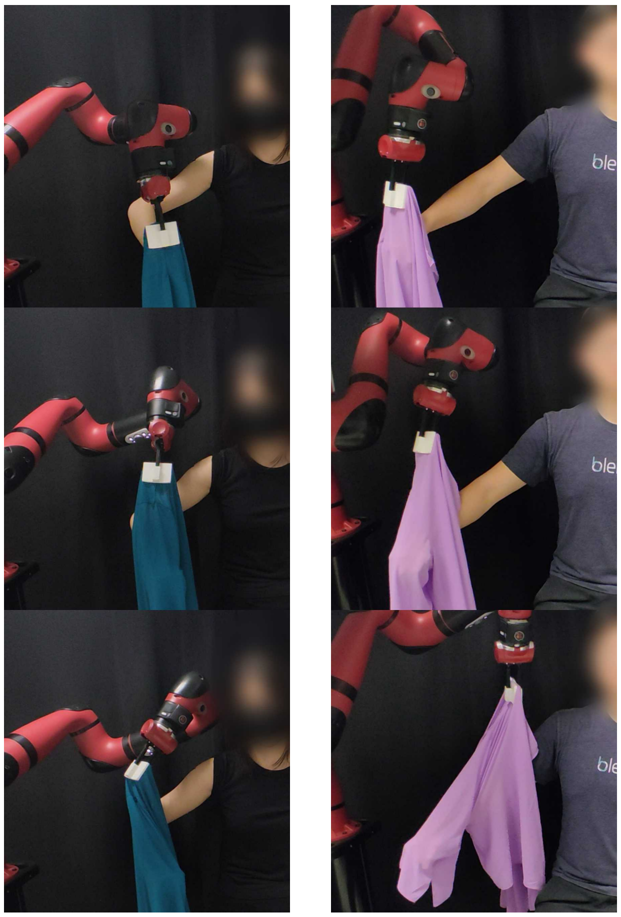


Figure A.5: Failure cases of our system in the human study. Left: the policy gets stuck and stops output actions that make any further progress of the task. Right: the garment gets caught on the participant because the policy actions pull the gripper to high above the participant's arm.

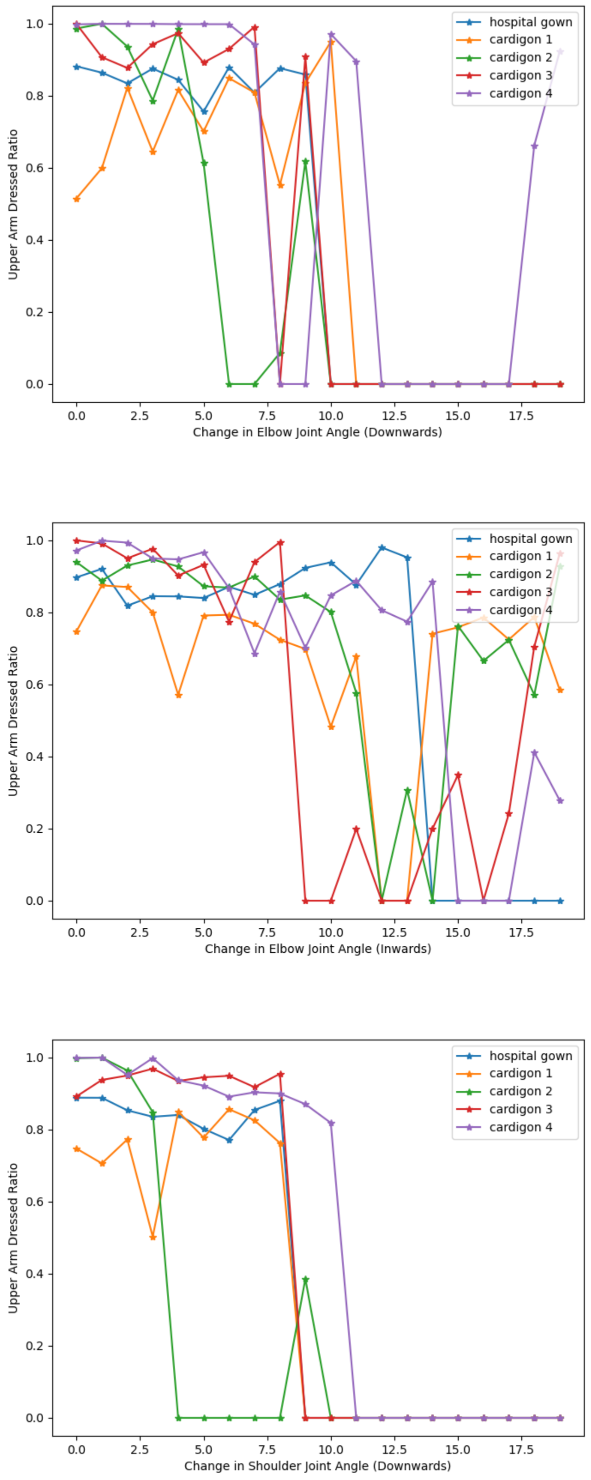


Figure A.6: Upper arm dressed ratio versus changes in arm joint angles.

A.3.3 Failure Cases

Figure A.5 shows two failure cases from our policy. The first failure case is that the policy gets stuck and cannot output actions that make further progress for the task, e.g., the policy oscillates between moving up and down and does not move forward. This might be due to the gap in sim2real transfer or out-of-distribution arm poses/sizes/shapes in the real world. The second failure case occurs when the garment gets caught on the participant. Since it is visually difficult to detect if the garment has gotten stuck, incorporating force-torque sensing could be helpful in addressing this failure case.

A.4 Out of Distribution Evaluation

As mentioned in the main paper, we perform evaluation of our system when the static arm assumption is relaxed. In simulation, we evaluate how our system performs if the participants change their shoulder or elbow joint angles after we capture the initial arm point cloud, on 1 pose sub-region and 5 garments. We test 3 types of joint angle changes: lowering down the shoulder joint, lowering down the elbow joint, and bending inwards the elbow joint. Figure A.6 demonstrates how the upper arm dressed ratio varies as the change of joint angle varies. Overall, we find our system to be robust to 8.6 degrees of change in shoulder and elbow joint angles (averaged across 3 types of joint angle changes and 5 garments) while maintaining 75% of the original performance.

Appendix B

RL-VLM-F

B.1 Details on Tasks and Environments

We run our method and baselines on *CartPole* from openAI Gym [15], three rigid and articulated object manipulation tasks from MetaWorld [122], and three deformable object manipulation tasks from SoftGym [65]. For the three MetaWorld tasks, we modified the gripper initial state such that it starts close to the target object to manipulate. Figure 3 in the paper shows the initial state for these 3 tasks. We also adjusted the camera view such that the target object is clearly visible at around the center of the image, to provide good images for VLM to give preferences. We describe the observation space and action space for those tasks as follows:

B.1.1 Observation Space

For policy learning with SAC, we use state-based observations; for reward learning, we use high dimensional RGB image observations, rendered by the simulator. We now detail the state-based observation space for each task.

MetaWorld Tasks. For MetaWorld tasks, we follow the setting in the original paper [122]. The state observation always has 39 dimensions. It consists of the position and gripper status of the robot's end-effector, the position and orientation of objects in the scene, and the position of the goal.

CartPole. The state observation has 4 dimensions, including the position and velocity of the cart, as well as the angle and angular velocity of the pole.

Cloth Fold. The state observation is the position of a subset of the particles in the cloth mesh. The cloth is of size 40 x 40, and we uniformly subsample it to be of size 8 x 8. The state is then the position of the picker, and the positions of all those subsampled particles.

Straighten Rope. The state observation is the positions of all particles on the rope and has 36 dimensions.

Pass Water. The state observation includes the size (width, length, height) of the container, the target container position, height of the water in the container, amount of water inside and outside of the container. The state observation has 7 dimensions.

B.1.2 Action Space

For all environments, we normalize the action space to be within $[-1, 1]$. Below we describe the action space for each environment.

MetaWorld Tasks. For MetaWorld tasks, the action space always has four dimensions. It includes the change in 3D position of the robot's end-effector followed by a normalized torque that the gripper fingers should apply.

CartPole. The original action space is a discrete value in $0, 1$, indicating the direction of the fixed force the cart is pushed with. We modified it to be continuous within range [0, 1] such that SAC can be used as the learning algorithm. The continuous action represents the force applied to the pole.

Cloth Fold. For this task, we use a pick-and-place action primitive. We assume that the corner of the cloth is grasped when the task is initialized. The action is the 2D target place location.

Straighten Rope. For this task, we use two pickers, one at each end of the rope, to control the rope. Therefore, the action space is the 3D delta positions for each picker and has 6 dimensions in total. We assume the two end points of the rope is already grasped at the beginning of the task.

Pass Water. The motion of the glass container is constrained to be in one dimension. Therefore, the action also has a dimension of 1 and is the delta position of the container along the dimension.

B.2 Hyper-parameters and Network Architectures

B.2.1 Image-based Reward Learning

For the image-based reward model, we use a 4-layer Convolutional Neural Network for MetaWorld tasks and *CartPole* and a standard ResNet-18 [42] for the three deformable object manipulation tasks. Following PEBBLE [56], we also use an ensemble of three reward models and use tanh as the activation function for outputting reward. For RL-VLM-F, we train the model by optimizing the cross-entropy loss, defined in Equation 2. For VLM Score, we train the mode by optimizing the MSE loss between the predicted score and ground-truth score output by the VLM. For both methods, we use ADAM [51] as the optimizer with an initial learning rate of 0.0003.

B.2.2 Policy Learning

Following PEBBLE [56], we use SAC as the off-policy learning algorithm. We follow the network architectures for the actor and critic and all the hyper-parameter settings in the original paper for policy learning.

B.2.3 Training details

Our implementation is based on PEBBLE [56]. Below we describe the feedback collection schedule for each task. For all tasks, we use a segment size of 1. We summarize the number of queries per feedback session (M in Algorithm 1), the frequency at which we collect feedback in terms of environment steps (K in Algorithm 1), and the maximum budget of queries (N) for each task in Table B.1. For Cloth Fold, we have to use a small number of maximum budget of queries due to the quota limitation of GPT-4V.

	M	K	N
Open Drawer	40	4000	20000
Soccer	40	4000	20000
Sweep Into	40	4000	20000
CartPole	50	5000	10000
Cloth Fold	50	1000	500
Straighten Rope	100	5000	12000
Pass Water	100	5000	12000

Table B.1: Hyper-parameters for feedback learning schedule.

B.3 Baselines

B.3.1 VLM score

For this baseline, we use the same amount of queries (K) at the same frequency (M) as in our method to ask VLM to directly output a score between 0 to 1. The reward model's architecture is the same as our method, except that the model is trained with regression loss to regress to VLM's output score instead of classification loss as done in our method.

B.3.2 RoboCLIP

In RoboCLIP, the backbone video-language model is S3D [117], trained on clips of human activities paired with textual descriptions from the HowTo100M dataset [74]. Given the assumption that the model generalizes to unseen robotic environments, we applied this baseline solely to the three MetaWorld tasks that contain a robot in the scene. We obtain the implementation directly from the authors. To maintain uniform assumptions across methods, we compare against the RoboCLIP variant that only uses a text description instead of a video demonstration to compute the similarity score with the agent's episode rollout for reward computation. According to the original paper, this text-only variant of RoboCLIP underperforms the video-based method, corroborating the lower performance observed in our tasks.

Prompt Template for RL-VLM-F (ours)

Analysis Template

Consider the following two images:
Image 1:

[Image 1]
Image 2:

[Image 2]

1. What is shown in Image 1?
2. What is shown in Image 2?

3. The goal is to [task description]. Is there any difference between Image 1 and Image 2 in terms of achieving the goal?

Labeling Template

Based on the text below to the questions:
[Repeat the 3 questions in the Analysis Template]

[VLM response]
Is the goal better achieved in Image 1 or Image 2? Reply a single line of 0 if the goal is better achieved in Image 1, or 1 if it is better achieved in Image 2.
Reply -1 if the text is unsure or there is no difference.

Figure B.1: Prompt Template for RL-VLM-F.

B.4 Prompts

B.4.1 RL-VLM-F and VLM Score

For both RL-VLM-F and VLM Score, we use a unified query template combined with specific task goal descriptions. The templates for RL-VLM-F and VLM Score are shown in Figure B.1 and Figure B.2:

The only task-specific part in both prompts is the task goal description. We use the same set of descriptions for both methods. We summarize the textual description for each task in Table B.2.

Prompt Template for VLM Score

Analysis Template Consider the following image:
[Image]

1. What is shown in the image?
2. The goal is [task description]. On a scale of 0 to 1, the score is 1 if the goal is achieved. What score would you give the image in terms of achieving the goal?

Labeling Template

Based on the text below to the questions:
[Repeat the 3 questions in the Analysis Template]

[VLM response]
Please reply a single line of the score the text has given. Reply -1 if the text is unsure.

Figure B.2: Prompt Template for VLM Score.

B.4.2 CLIP Score and BLIP-2 Score

The task descriptions for both CLIP Score and BLIP-2 Score baselines are summarized in Table B.3. The semantic meaning is almost identical to those used by RL-VLM-F and VLM Score, except that the description is structured differently. For *CartPole*, we used the exact same prompt as in [92], since they reported successful learning of this task using that prompt.

B.4.3 RoboCLIP

For the task descriptions for the RoboCLIP baseline, we followed the format used in the original paper [99]. We summarize the text descriptions in Table B.4.

B.4.4 RL-VLM-F single stage prompt

In Section 3.1.6 we compared to an ablated version of RL-VLM-F where a single-stage prompting strategy is used. The single-stage prompt used is shown in Figure B.3. For a fair comparison, it is kept to be the same as the two-stage prompt with only

Single Stage Prompt Template for RL-VLM-F

Consider the following two images:
Image 1:

[Image 1]
Image 2:

[Image 2]

1. What is shown in Image 1?
2. What is shown in Image 2?

3. The goal is [task description]. Is there any difference between Image 1 and Image 2 in terms of achieving the goal?

Is the goal better achieved in Image 1 or Image 2? Reply a single line of 0 if the goal is better achieved in Image 1, or 1 if it is better achieved in Image 2.

Reply -1 if the text is unsure or there is no difference.

Figure B.3: The single stage prompt Template for RL-VLM-F.

minor differences.

B.5 Additional Experiment Results

B.5.1 GT Task Reward (Oracle) and GT Sparse Reward (Oracle)

To better contextualize the results from different reward models, we test two more baselines, i.e., GT Task Reward (Oracle) and GT Sparse Reward (Oracle). For GT Task Reward (Oracle), we use the original ground-truth human-written reward function with SAC as the RL algorithm to train the policy. For GT Sparse Reward (Oracle), we use sparse reward with SAC. The reward is 1 when the goal is achieved and 0 otherwise. The results of GT Task Reward (Oracle) and GT Sparse Reward (Oracle), along with our method and all baselines, are shown in Figure B.4. For most

Task Name	Goal Description
Open Drawer	to open the drawer
Soccer	to move the soccer ball into the goal
Sweep Into	to minimize the distance between the green cube and the hole
CartPole	to balance the brown pole on the black cart to be upright
Cloth Fold	to fold the cloth diagonally from top left corner to bottom right corner
Straighten Rope	to straighten the blue rope
Pass Water	to move the container, which holds water, to be as close to the red circle as possible without causing too many water droplets to spill

Table B.2: Goal description used in RL-VLM-F and VLM Score baseline.

Task Name	Goal Description
Open Drawer	The drawer is opened.
Soccer	The soccer ball is in the goal.
Sweep Into	The green cube is in the hole.
CartPole	Pole vertically upright on top of the cart.
Cloth Fold	The cloth is folded diagonally from top left corner to bottom right corner.
Straighten Rope	The blue rope is straightened.
Pass Water	The container, which holds water, is as close to the red circle as possible without causing too many water droplets to spill.

Table B.3: Goal description used in CLIP Score and BLIP-2 Score.

Task Name	Goal Description
Open Drawer	Robot opening green drawer
Soccer	Robot pushing the soccer ball into the goal
Sweep Into	Robot sweeping the green cube into the hole on the table

Table B.4: Goal description used in RoboCLIP.

tasks, RL-VLM-F ’s final performance can match that of using ground-truth reward, highlighting the effectiveness of our method.

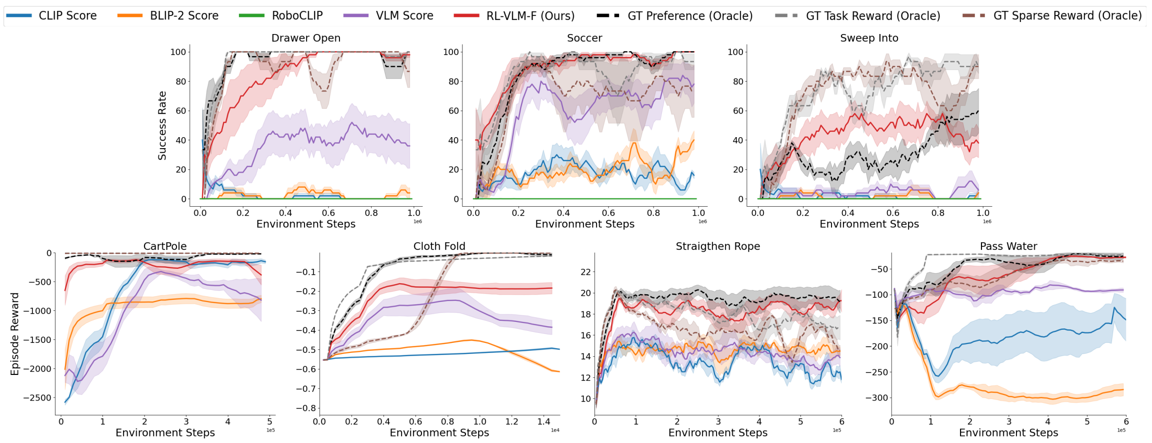


Figure B.4: Learning curves of GT Task Reward (Oracle) and GT Sparse Reward (Oracle), along with RL-VLM-F and all baselines.

B.5.2 Ablation Study: Influence of Using Different VLMs

For RL-VLM-F and the VLM score baseline, we use Gemini-Pro [103] as the VLM for all tasks except Fold Cloth. We find Gemini-Pro to perform poorly on Fold Cloth, so we instead use GPT-4V [80] as the VLM for this task for both methods. Figure B.5 compares the learning performance of Gemini-Pro versus GPT-4V on the task of Fold Cloth. We do observe GPT-4V to achieve much better performance on this task than Gemini-Pro. The poorer performance of Gemini-Pro on this task could be possibly due to the more complex visual reasoning required for deformable cloth.

B.5.3 More Visualization of the Learned Reward

Here we show the learned reward from RL-VLM-F and the VLM Score baseline, as well as the CLIP and BLIP-2 score along an expert trajectory on three MetaWorld tasks. We compare the learned reward from RL-VLM-F and the VLM Score / CLIP and BLIP-2 score to the ground-truth task progress. The results are shown in Figure B.6. For all three tasks, the reward learned by RL-VLM-F aligns the best with the ground-truth task progress.

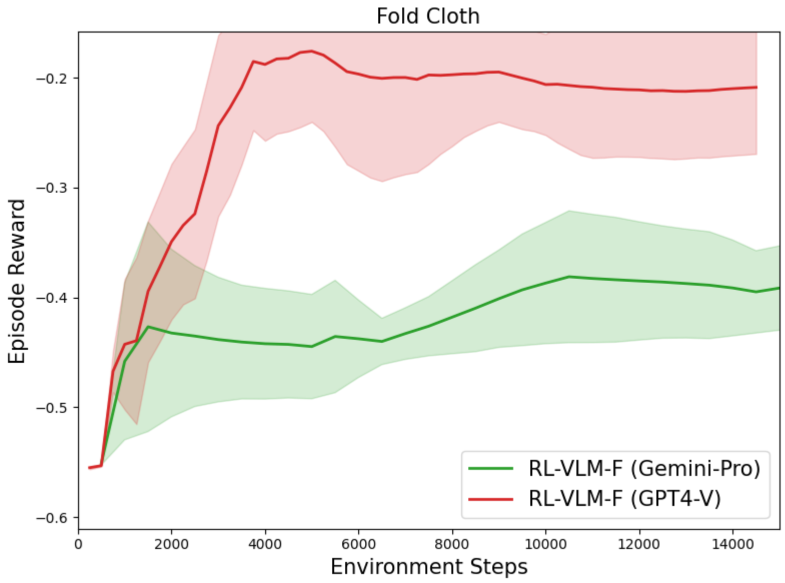


Figure B.5: On the Fold Cloth task, we find the performance of GPT-4V to be better than Gemini-Pro, possibly due to the complex visual appearance of the cloth.

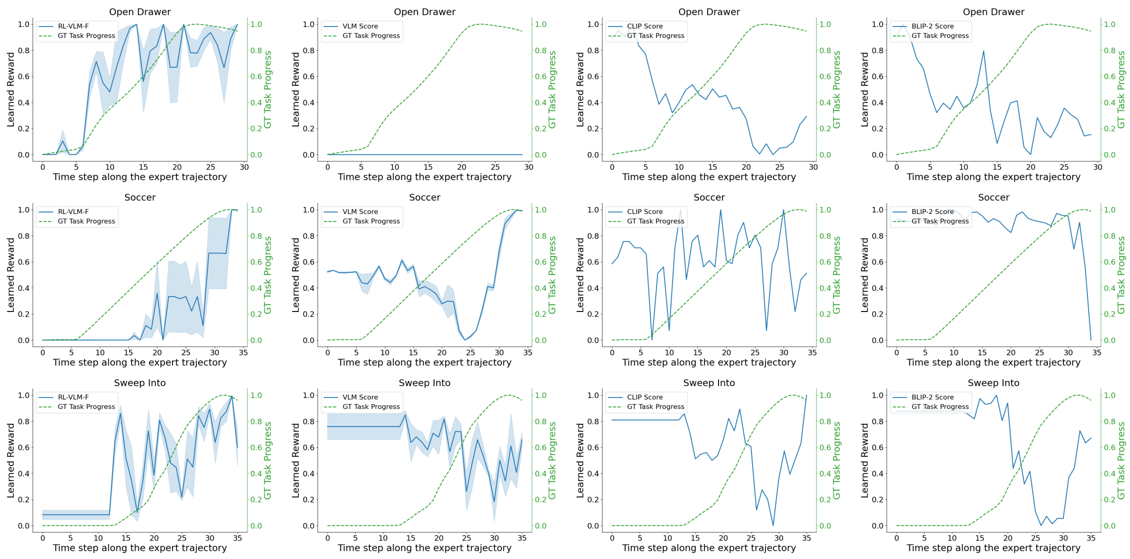


Figure B.6: Comparison of learned reward functions from RL-VLM-F and VLM Score, as well as CLIP and BLIP-2 score to the ground-truth task progress along a trajectory rollout on three MetaWorld tasks. From left column to right: reward learned by RL-VLM-F, reward learned by VLM Score, CLIP Score, BLIP-2 Score. From top row to bottom: *Open Drawer*, *Soccer*, and *Sweep Into*. The reward learned by RL-VLM-F aligns the best across all compared methods.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015430. 3.1.2

[2] Ademi Adeniji, Amber Xie, Carmelo Sferrazza, Younggyo Seo, Stephen James, and Pieter Abbeel. Language reward modulation for pretraining reinforcement learning, 2023. 3.1.1, 3.1.6

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 3.1.7

[4] Yahav Avigal, Lars Berscheid, Tamim Asfour, Torsten Kröger, and Ken Goldberg. Speedfolding: Learning efficient bimanual folding of garments. *arXiv preprint arXiv:2208.10552*, 2022. 2.1.2

[5] Arpit Bahety, Shreeya Jain, Huy Ha, Nathalie Hager, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Bag all you need: Learning a generalizable bagging strategy for heterogeneous objects. *arXiv preprint arXiv:2210.09997*, 2022. 2.1.2

[6] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022. 3.1.6

[7] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. 3.1.2

[8] Cristian C Beltran-Hernandez, Damien Petit, Ixchel G Ramirez-Alpizar, and Kensuke Harada. Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach. *Applied Sciences*, 10(19):6923, 2020. 2.2.2

[9] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Cloth3d: clothed 3d humans. In *European Conference on Computer Vision*, pages 344–359. Springer, 2020. 2.1.5, 2.2.6, A.2.1

[10] Homanga Bharadhwaj, Abhinav Gupta, Vikash Kumar, and Shubham Tulsiani. Towards generalizable zero-shot manipulation via translating human interaction plans. *arXiv preprint arXiv:2312.00775*, 2023. 3.1.6

[11] Erdem Biyik, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, 2019. 3.1.2

[12] Erdem Biyik, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. Active preference-based gaussian process regression for reward learning. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2020. doi: 10.15607/rss.2020.xvi.041. 3.1.2

[13] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022. 3.1.2

[14] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 3.1.3

[15] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 3.1.6, B.1

[16] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, 2018. 2.2.2

[17] Gerard Canal, Guillem Alenyà, and Carme Torras. Adapting robot task planning to user preferences: an assistive shoe dressing example. *Autonomous Robots*, 43 (6):1343–1356, 2019. 1.1, 2.1.2

[18] Alper Canberk, Cheng Chi, Huy Ha, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Cloth funnels: Canonicalized-alignment for multi-purpose garment manipulation. *arXiv preprint arXiv:2210.09347*, 2022. 2.1.4

[19] Lawrence Yunliang Chen, Baiyu Shi, Daniel Seita, Richard Cheng, Thomas Kollar, David Held, and Ken Goldberg. Autobag: Learning to open plastic bags and insert objects. *arXiv preprint arXiv:2210.17217*, 2022. 2.1.2

[20] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 3.1.1, 3.1.2, 3.1.3

[21] Kun Chu, Xufeng Zhao, Cornelius Weber, Mengdi Li, and Stefan Wermter. Accelerating reinforcement learning of robotic manipulations via feedback from large language models. *arXiv preprint arXiv:2311.02379*, 2023. 3.1.1

[22] 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. 2.1.2

[23] Yuchen Cui, Scott Niekum, Abhinav Gupta, Vikash Kumar, and Aravind Rajeswaran. Can foundation models perform zero-shot task specification for robot manipulation? *Learning for Dynamics and Control Conference*, 2022. 3.1.2

[24] Yuchen Cui, Scott Niekum, Abhinav Gupta, Vikash Kumar, and Aravind Rajeswaran. Can foundation models perform zero-shot task specification for robot manipulation? In Roya Firoozi, Negar Mehr, Esen Yel, Rika Antonova, Jeannette Bohg, Mac Schwager, and Mykel Kochenderfer, editors, *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pages 893–905. PMLR, 23–24 Jun 2022. URL `https://proceedings.mlr.press/v168/cui22a.html`. 3.1.1, 3.1.6

[25] Maximilian Du, Olivia Y Lee, Suraj Nair, and Chelsea Finn. Play it by ear: Learning skills amidst occlusion through audio-visual imitation learning. *arXiv preprint arXiv:2205.14850*, 2022. 2.2.2

[26] Ben Eisner, Harry Zhang, and David Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. *arXiv preprint arXiv:2205.04382*, 2022. 2.1.2, 2.1.4

[27] 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. 1.1, 2.1.2

[28] 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. 2.1.1, 2.1.2, 2.1.3, 2.1.5, 2.2.1, 2.2.6, A.2.2

[29] Zackory Erickson, Henry M Clever, Vamsee Gangaram, Greg Turk, C Karen

Liu, and Charles C Kemp. Multidimensional capacitive sensing for robot-assisted dressing and bathing. In *2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR)*, pages 224–231. IEEE, 2019. 2.1.1

[30] Zackory Erickson, Henry M Clever, Vamsee Gangaram, Eliot Xing, Greg Turk, C Karen Liu, and Charles C Kemp. Characterizing multidimensional capacitive servoing for physical human-robot interaction. *IEEE Transactions on Robotics (T-RO)*, 2022. 2.2.3, 2.2.6

[31] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 2022. 3.1.2

[32] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. A.1.3

[33] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 3.1.2

[34] 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. 1.1, 2.1.1, 2.1.2, 2.2.1, 2.2.6

[35] 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. 1.1, 2.1.1, 2.1.2, 2.2.1

[36] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015. 2.2.2

[37] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham M. Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity, 2022. 3.1.1

[38] 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. 2.1.2, 2.1.4

[39] 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. 2.1.4, 2.2.4, 3.1.5, A.1.3

[40] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022. 2.1.5, 2.1.5

[41] Lauren D Harris-Kojetin, Manisha Sengupta, Jessica Penn Lendon, Vincent Rome, Roberto Valverde, and Christine Caffrey. Long-term care providers and services users in the united states, 2015-2016. 2019. 2.1.1, 2.2.1

[42] 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. B.2.1

[43] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016. 3.1.2

[44] 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, 2022. 2.1.2

[45] Wenlong Huang, Igor Mordatch, Pieter Abbeel, and Deepak Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021. 2.1.2

[46] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018. 3.1.2, 3.1.3

[47] Pedro Ildefonso, Pedro Remédios, Rui Silva, Miguel Vasco, Francisco S Melo, Ana Paiva, and Manuela Veloso. Exploiting symmetry in human robot-assisted dressing using reinforcement learning. In *EPIA Conference on Artificial Intelligence*, pages 405–417. Springer, 2021. 2.1.1, 2.1.2

[48] Aleksandar Jevtić, Andrés Flores Valle, Guillem Alenyà, Greg Chance, Praminda Caleb-Solly, Sanja Dogramadzi, and Carme Torras. Personalized robot assistant for support in dressing. *IEEE transactions on cognitive and developmental systems*, 11(3):363–374, 2018. 1.1, 2.1.2

[49] 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. 2.1.2

[50] Ariel Kapusta, Zackory Erickson, Henry M Clever, Wenhao Yu, C Karen Liu, Greg Turk, and Charles C Kemp. Personalized collaborative plans for robot-assisted dressing via optimization and simulation. *Autonomous Robots*, 43(8):

2183–2207, 2019. 1.1, 2.1.1, 2.1.2, 2.1.3, 2.1.5, 2.2.1, A.2.2

[51] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. A.1.3, A.2.2, B.2.1

[52] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2.1.6

[53] Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023. 3.1.1, 3.1.2

[54] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *International Conference on Learning Representations (ICLR)*, 2023. 3.1.2

[55] Adam Daniel Laud. *Theory and application of reward shaping in reinforcement learning*. PhD thesis, USA, 2004. AAI3130966. 3.1.1

[56] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training, 2021. 3.1.1, 3.1.2, 3.1.3, 3.1.3, 3.1.5, B.2.1, B.2.2, B.2.3

[57] Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*, 2021. 3.1.3

[58] 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. 2.2.2

[59] 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. 2.1.2

[60] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018. 3.1.2

[61] Ljung Lennart. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ*, 28:540, 1999. 4

[62] Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. See, hear, and feel:

Smart sensory fusion for robotic manipulation. *arXiv preprint arXiv:2212.03858*, 2022. 2.2.2

[63] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023. 3.1.6

[64] 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. 1.1, 2.1.2

[65] 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. 2.1.2, 2.1.5, 2.2.6, 3.1.6, A.2.1, B.1

[66] Xingyu Lin, Carl Qi, Yunchu Zhang, Zhiao Huang, Katerina Fragkiadaki, Yunzhu Li, Chuang Gan, and David Held. Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. In *6th Annual Conference on Robot Learning*, 2022. 2.1.2

[67] 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. 2.1.2

[68] Minghua Liu, Xuanlin Li, Zhan Ling, Yangyan Li, and Hao Su. Frame mining: a free lunch for learning robotic manipulation from 3d point clouds. *arXiv preprint arXiv:2210.07442*, 2022. 2.1.2, 2.1.4

[69] Zuxin Liu, Zhepeng Cen, Vladislav Isenbaev, Wei Liu, Steven Wu, Bo Li, and Ding Zhao. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*, pages 13644–13668. PMLR, 2022. 2.2.2, 2.2.6

[70] Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. *arXiv preprint arXiv:2306.00958*, 2023. 1.1, 3.1.1, 3.1.2

[71] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv: Arxiv-2310.12931*, 2023. 3.1.1, 3.1.2

[72] Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Viktor Makoviychuk. Non-smooth newton methods for deformable multi-body dynamics. *ACM Transactions on Graphics (TOG)*, 38(5):1–20, 2019. 2.2.6

[73] Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. Zero-shot reward specification via grounded natural language. In *International Conference on Machine Learning*, pages 14743–14752. PMLR, 2022. 3.1.1, 3.1.6

[74] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *ICCV*, 2019. B.3.2

[75] Vivek Myers, Erdem Biyik, Nima Anari, and Dorsa Sadigh. Learning multimodal rewards from rankings. In *5th Annual Conference on Robot Learning*, 2021. 3.1.2

[76] Taewook Nam, Juyong Lee, Jesse Zhang, Sung Ju Hwang, Joseph J. Lim, and Karl Pertsch. Lift: Unsupervised reinforcement learning with foundation models as teachers, 2023. 1.1, 3.1.2

[77] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072. 3.1.2

[78] Tianwei Ni, Harshit Sikchi, Yufei Wang, Tejus Gupta, Lisa Lee, and Ben Eysenbach. f-irl: Inverse reinforcement learning via state marginal matching. In *Conference on Robot Learning*, pages 529–551. PMLR, 2021. 3.1.2

[79] Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. *arXiv preprint arXiv:2301.12050*, 2023. 3.1.2

[80] OpenAI. Gpt-4v(ision) system card. 2023. 3.1.1, 3.1.4, 3.1.6, B.5.2

[81] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019. 3.1.1

[82] Chuer Pan, Brian Okorn, Harry Zhang, Ben Eisner, and David Held. Tax-pose: Task-specific cross-pose estimation for robot manipulation. *arXiv preprint arXiv:2211.09325*, 2022. 2.1.2

[83] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015. 2.1.5

[84] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019. 2.1.5, 2.2.6, A.2.1

[85] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018. 4

[86] Kavya Puthuveetil, Charles C Kemp, and Zackory Erickson. Bodies uncovered: Learning to manipulate real blankets around people via physics simulations. *IEEE Robotics and Automation Letters*, 7(2):1984–1991, 2022. 2.1.2

[87] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2.1.4, 2.2.5

[88] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2.1.4, 2.2.4, A.1.3

[89] Jinge Qie, Yixing Gao, Runyang Feng, Xin Wang, Jielong Yang, Esha Dasgupta, Hyung Jin Chang, and Yi Chang. Cross-domain representation learning for clothes unfolding in robot-assisted dressing. In *Tenth International Workshop on Assistive Computer Vision and Robotics*, 2022. 2.1.1, 2.1.2, 2.1.3

[90] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. *arXiv preprint arXiv:2211.09423*, 2022. 2.1.2, 2.1.4

[91] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 3.1.1, 3.1.6

[92] Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL `https://openreview.net/forum?id=JUwczEJY8I`. 1.1, 3.1.1, 3.1.2, 3.1.6, B.4.2

[93] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Ko-

ray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015. 2.1.4, 2.1.5

[94] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018. 2.1.4, 2.1.4, 2.1.5

[95] 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. 2.1.2

[96] 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. 2.1.2

[97] Daniel Seita, Yufei Wang, Sarthak J Shetty, Edward Yao Li, Zackory Erickson, and David Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. *arXiv preprint arXiv:2211.09006*, 2022. 2.1.2, 2.1.4

[98] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 3.1.1

[99] Sumedh Anand Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=DVlawv2rSI`. 1.1, 3.1.1, 3.1.2, 3.1.6, B.4.3

[100] Priya Sundaresan, Suneel Belkhale, and Dorsa Sadigh. Learning visuo-haptic skewering strategies for robot-assisted feeding. In *6th Annual Conference on Robot Learning*, 2022. 2.2.2

[101] Neha Sunil, Shaoxiong Wang, Yu She, Edward Adelson, and Alberto Rodriguez Garcia. Visuotactile affordances for cloth manipulation with local control. In *Conference on Robot Learning*, pages 1596–1606. PMLR, 2023. 2.2.2

[102] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 3.1.3

[103] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja

Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 3.1.1, 3.1.4, 3.1.6, B.5.2

[104] 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. 4

[105] Lirui Wang, Yu Xiang, Wei Yang, Arsalan Mousavian, and Dieter Fox. Goal-auxiliary actor-critic for 6d robotic grasping with point clouds. In *Conference on Robot Learning*, pages 70–80. PMLR, 2022. 2.1.2, 2.1.4

[106] Yufei Wang, David Held, and Zackory Erickson. Visual haptic reasoning: Estimating contact forces by observing deformable object interactions. *IEEE Robotics and Automation Letters*, 7(4):11426–11433, 2022. 1.1, 2.1.2

[107] Yufei Wang, Zhanyi Sun, Zackory Erickson, and David Held. One policy to dress them all: Learning to dress people with diverse poses and garments. In *Robotics: Science and Systems (RSS)*, 2023. 2.2.1, 2.2.4, 2.2.6, 2.2.6, **??**, 2.2.6

[108] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023. 3.1.1, 3.1.2

[109] David Watkins-Valls, Jacob Varley, and Peter Allen. Multi-modal geometric learning for grasping and manipulation. In *2019 International conference on robotics and automation (ICRA)*, pages 7339–7345. IEEE, 2019. 2.2.2

[110] 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. 2.1.2

[111] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3583–3590. IEEE, 2022. 2.2.2

[112] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017. URL `http://jmlr.org/papers/v18/16-634.html`. 3.1.1, 3.1.2

[113] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Johnny Lee, Szymon Rusinkiewicz, and Thomas Funkhouser. Spatial action maps for mobile manipulation. *arXiv preprint arXiv:2004.09141*, 2020. 2.1.4

[114] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Szymon Rusinkiewicz, and Thomas Funkhouser. Spatial intention maps for multi-agent mobile manip-

ulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8749–8756. IEEE, 2021. 2.1.4

[115] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019. 2.1.2

[116] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. *arXiv preprint arXiv:2204.02320*, 2022. 2.1.2

[117] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 305–321, 2018. 3.1.6, B.3.2

[118] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023. 3.1.1, 3.1.2

[119] Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Dextairity: Deformable manipulation can be a breeze. *arXiv preprint arXiv:2203.01197*, 2022. 2.1.2

[120] Xitong Yang, Palghat Ramesh, Radha Chitta, Sriganesh Madhvanath, Edgar A Bernal, and Jiebo Luo. Deep multimodal representation learning from temporal data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5447–5455, 2017. 2.2.2

[121] 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. 2.1.5, 2.1.5, A.2.2

[122] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020. 3.1.6, 3.1.6, B.1, B.1.1

[123] 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. 1.1, 2.1.2

[124] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Lan-

guage to rewards for robotic skill synthesis. *Arxiv preprint arXiv:2306.08647*, 2023. 3.1.2

[125] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018. 2.1.4

[126] 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. 2.1.4

[127] Fan Zhang and Yiannis Demiris. Learning grasping points for garment manipulation in robot-assisted dressing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9114–9120. IEEE, 2020. 2.1.2, 2.1.3, 2.2.3

[128] Fan Zhang and Yiannis Demiris. Learning garment manipulation policies toward robot-assisted dressing. *Science robotics*, 7(65):eabm6010, 2022. 2.1.2, 2.2.1, 2.2.3

[129] 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. 1.1, 2.1.2, 2.2.2

[130] 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. 1.1, 2.1.1, 2.1.2, 2.2.1, 2.2.2, 2.2.6

[131] Jihong Zhu, Andrea Cherubini, Claire Dune, David Navarro-Alarcon, Farshid Alambeigi, Dmitry Berenson, Fanny Ficuciello, Kensuke Harada, Jens Kober, Xiang Li, et al. Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 29(3):67–77, 2022. 2.1.1

[132] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1433–1438. AAAI Press, 2008. ISBN 9781577353683. 3.1.2