Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Learning to Manipulate Deformable Objects without Demonstrations

Yilin Wu*, Wilson Yan*, Thanard Kurutach, Lerrel Pinto, Pieter Abbeel
University of California, Berkeley
{yilin-wu,wilson1.yan}@berkeley.edu

***Abstract*—In this paper we tackle the problem of deformable object manipulation through model-free visual reinforcement learning (RL). In order to circumvent the sample inefficiency of RL, we propose two key ideas that accelerate learning. First, we propose an iterative pick-place action space that encodes the conditional relationship between picking and placing on deformable objects. The explicit structural encoding enables faster learning under complex object dynamics. Second, instead of jointly learning both the pick and the place locations, we only explicitly learn the placing policy conditioned on random pick points. Then, by selecting the pick point that has Maximal Value under Placing (MVP), we obtain our picking policy. This provides us with an informed picking policy during testing, while using only random pick points during training. Experimentally, this learning framework obtains an order of magnitude faster learning compared to independent action-spaces on our suite of deformable object manipulation tasks with visual RGB observations. Finally, using domain randomization, we transfer our policies to a real PR2 robot for challenging cloth and rope coverage tasks, and demonstrate significant improvements over standard RL techniques on average coverage.**

I. Introduction

Over the last few decades, we have seen tremendous progress in robotic manipulation. From grasping objects in clutter [56, 44, 31, 27, 12] to dexterous in-hand manipulation of objects [1, 69], modern robotic algorithms have transformed object manipulation. But much of this success has come at the price of making a key assumption: rigidity of objects. Most robot algorithms often require (implicitly or explicitly) strict rigidity constraints on objects. But the objects we interact with everyday, from the clothes we put on to shopping bags we pack, are deformable. In fact, even 'rigid' objects deform under different form factors (like a metal wire). Because of this departure from the 'rigid-body' assumption, several real-world applications of manipulation fail [20]. So why haven't we created equally powerful algorithms for deformable objects yet?

Deformable object manipulation has been a long standing problem [65, 15, 59, 32, 54], with two unique challenges. First, in contrast with rigid objects, there is no obvious representation of state. Consider the cloth manipulation problem in Fig. 1(a), where the robot needs to flatten a cloth from any start configuration. How do we track the shape of the cloth? Should we use a raw point cloud, or fit a continuous function? This lack of canonical state often limits state representations to discrete approximations [2]. Second, the dynamics is complex and non-linear [7]. Due to microscopic interactions in the object, even simple looking objects can exhibit complex and unpredictable behavior [43]. This makes it difficult to model and perform traditional task and motion planning.

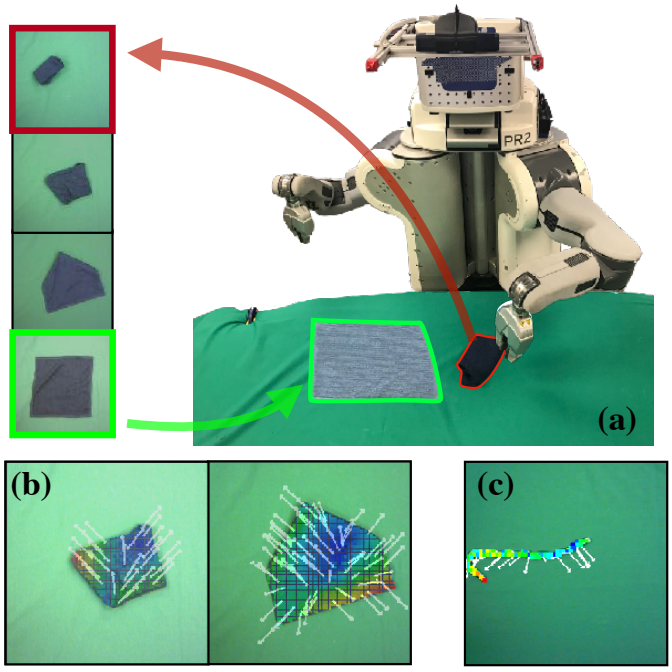

Fig. 1: We look at the problem of deformable object manipulation, where the robot needs to manipulate a deformable object, say the blue cloth, into a desired goal location (green in (a)). Our method learns an explicit placing policy (arrows in (b) and (c)), along with an implicit picking policy. This method is evaluated on cloth (b) and rope (c) tasks using our PR2 robot. The heatmaps represent the distribution of the Q-value, where the Q-values over each pick location are normalized to the range of 0 (blue) to 1 (red).

One of the recent breakthroughs in robotics has been the development of model-free visual policy learning [26, 45, 1], where robotic algorithms can reason about interactions directly from raw sensory observations. This can alleviate the challenge of state estimation for deformable objects [33], since we can directly learn on images. Moreover, since these methods do not require an explicit model of the object [29], they can overcome the challenge of having complex deformable object dynamics. But model-free learning has notoriously poor sample complexity [6]. This has limited the application of learning to the setting where human demonstrations are available [41, 33]. To

reduce the dependence on human demonstrators, Seita et al. [55] in concurrent and independent work, has shown how expert state-based policies can provide simulated demonstrations to learn cloth manipulation from visual observations.

In this work, we tackle the sample-complexity issue by focusing on an often ignored aspect of learning: the action space. Inspired by Howard and Bekey [16], Brooks [4], we start by using an iterative pick-place action space, where the robot can decide which point to grasp (or pick) and to which point it should drop (or place). But how should one learn with this action space? One option is to directly output both the pick point and place location for the deformable object. But the optimal placing location is heavily correlated with picking location, i.e. where you place depends heavily on what point you pick. This conditional structure makes it difficult to simultaneously learn without modeling the action space.

To solve this, we propose a conditional action space, where the output of the picking policy is fed as input into the placing policy. This type of action space is inspired by recent work in auto-regressive output spaces in image generation [64], imitation learning [41], and grasping [62]. However in the context of model-free RL, this leads us to a second problem: the placing policy is constrained by the picking policy. When learning starts, the picking policy often collapses into a suboptimal restrictive set of pick points. This inhibits the exploration of the placing policy, since the picking points it takes as input are only from a restrictive set, and results in a suboptimal placing policy. Now, since the rewards for picking come after the placing is executed, the picking policy receives poor rewards and results in inefficient learning. This illustrates the chicken and egg problem with conditional action spaces. Learning a good picking strategy involves having a good placing strategy, while learning a good placing strategy involves having a good picking strategy.

To break this chicken and egg loop, we learn the placing strategy independent of the picking strategy. This allows us to both learn the placing policy efficiently, and use the learned placing value approximator [29] to inform the picking policy. More concretely, since the value of the placing policy is conditioned on the pick point, we can find the pick point that maximizes the value. We call this picking policy Maximum Value of Placing (MVP). During training, the placing policy is trained with a random picking policy. However, during testing, the MVP picking policy is used. Through this, we observe a significant speedup in convergence on three difficult deformable object manipulation tasks on rope and cloth objects. Finally, we demonstrate how this policy can be transferred from a simulator to a real robot using simple domain randomization without any additional real-world training or human demonstrations. Videos of our PR2 robot performing deformable object manipulation along with our code can be accessed on the project website: https://sites.google.com/view/alternating-pick-and-place. Interestingly, our policies are able to generalize to a variety of starting states of both cloth and rope previously unseen in training.

In summary, we present three contributions in this paper: (a) we propose a novel learning algorithm for picking based on the maximal value of placing; (b) we show that the conditional action space formulation significantly accelerates the learning for deformable object manipulation; and (c) we demonstrate transfer to real-robot cloth and rope manipulation using our proposed formulation.

II. Related Work

A. Deformable Object Manipulation

Robotic manipulation of deformable objects has had a rich history that has spanned different fields from surgical robotics to industrial manipulation. For a more detailed survey, we refer the reader to Khalil and Payeur [23], Henrich and Wörn [15].

Motion planning has been a popular approach to tackle this problem, where several works combine deformable object simulations with efficient planning [19]. Early work [49, 66, 39] focused on using planning for linearly deformable objects like ropes. Rodriguez et al. [47] developed methods for fully deformable simulation environments, while Frank et al. [9] created methods for faster planning with deformable environments. One of the challenges of planning with deformable objects is the large degrees of freedom and hence large configuration space involved when planning. This, coupled with the complex dynamics [7], has prompted work in using high-level planners or demonstrations and local controllers to follow the plans.

Instead of planning on the full complex dynamics, we can plan on simpler approximations, but use local controllers to handle the actual complex dynamics. One way to use local controllers is model-based servoing [57, 65], where the end-effector is locally controlled to a given goal location instead of explicit planning. However, since the controllers are optimized over simpler dynamics, they often get stuck in local minima with more complex dynamics [36]. To solve this model-based dependency, Berenson [2], McConachie and Berenson [35], Navarro-Alarcon et al. [42] have looked at Jacobian approximated controllers that do not need explicit models, while Jia et al. [18], Hu et al. [17] have looked at learning-based techniques for servoing. However, since the controllers are still local in nature, they are still susceptible to reaching globally suboptimal policies. To address this, McConachie et al. [36] interleaves planning along with local controllers. Although this produces better behavior, transferring it to a robot involves solving the difficult state-estimation problem [51, 52]. Instead of a two step planner and local controller, we propose to directly use model-free visual learning, which should alleviate the state-estimation problem along with working with the true complex dynamics of the manipulated objects.

B. Reinforcement Learning for Manipulation

Reinforcement Learning (RL) has made significant progress in many areas including robotics. RL has enabled robots to handle unstructured perception such as visual inputs and reason about actions directly from raw observations [37], which can be desirable in many robotic tasks. RL from vision

has been shown to solve manipulation problems such as in-hand block manipulation [1, 46], object pushing [8], and valve-rotating with a three-fingered hand [14]. However, these algorithms have not yet seen wide applicability to deformable object manipulation. This is primarily due to learning being inefficient with complex dynamics [6], which we address in this work.

Over the last few years, deformable object manipulation has also been studied in reinforcement learning [41, 25, 33, 67, 55]. However, many of these works [25, 33] require expert demonstrations to guide learning for cloth manipulation. These expert demonstrations can also be used to learn wire threading [34, 50]. In concurrent work, Seita et al. [55] show that instead of human demonstrators, a simulated demonstrator using state information can be used to obtain demonstrations. Other works like Nair et al. [41] that do not need demonstrations for training require them at test time. Similarly, Wang et al. [67] do not use demonstration, but do require self-supervised exploration data at training time. We note that since using our conditional action spaces and MVP technique can be applied to any actor-critic algorithm, it is complementary to most methods that learn from expert demonstrations.

III. Background

Before we describe our learning framework, we briefly discuss relevant background on reinforcement learning and off-policy learning. For a more in-depth survey, we refer the reader to Sutton et al. [60], Kaelbling et al. [21].

A. Reinforcement Learning

We consider a continuous Markov Decision Process (MDP), represented by the tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, r, \gamma, s_0)$, with continuous state and action space, $\mathcal{S}$ and $\mathcal{A}$, and a partial observation space $\mathcal{O}$. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ defines the transition probability of the next state s_{t+1} given the current state-action pair (s_t, a_t). For each transition, the environment generates a reward $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$, with future reward discounted by γ.

Starting from an initial state s_0 sampled from distribution $\mathcal{S}$, the agent takes actions according to policy $\pi(a_t|s_t)$ and receives reward $r_t = r(s_t, a_t)$ at every timestep t. The next state s_{t+1} is sampled from the transition distribution $\mathcal{P}(s_{t+1}|s_t, a_t)$. The objective in reinforcement learning is to learn a policy that maximizes the expected sum of discounted rewards $\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi(s_t,a_t)}[\gamma^t r(s_t, a_t)]$. In the case of a partially observable model, the agent receives observations o_t and learns $\pi(a_t|o_t)$.

B. Off Policy Learning

On-policy reinforcement learning [53, 22, 68] iterates between data collection and policy updates, hence requiring new on-policy data per iteration which tends to be expensive to obtain. On the other hand, off-policy reinforcement learning retains past experiences in a replay buffer and is able to re-use past samples. Thus, in practice, off-policy algorithms have achieved significantly better sample efficiency [14, 24]. Off-policy learning can be divided into three main categories: model-based RL, Actor-Critic (AC), and Q learning. In model-based RL, we learn the dynamics of the system. In the AC framework, we learn both the policy (actor) and value function (critic). Finally, in Q-learning we often learn only the value function, and choose actions that maximize it.

In this work, we choose the actor-critic framework due to its stability, data-efficiency, and suitability for continuous control. Recent state-of-the-art actor-critic algorithms such as Twin Delayed DDPG (TD3) [10] and Soft-Actor-Critic (SAC) [13] show better performance than prior off-policy algorithms such Deep Deterministic Policy Gradient (DDPG) [28] and Asynchronous Advantage Actor-Critic (A3C) [38] due to variance reduction methods in TD3 by using a second critic network to reduce over-estimation of the value function and an additional entropy term in SAC to encourage exploration. In this work, we use SAC since its empirical performance surpasses TD3 (and other off-policy algorithms) on most RL benchmark environments [13]. However, our method is not tied to SAC and can work with any off-policy learning algorithm.

C. Soft Actor Critic

For our experiments, we use SAC [14], an entropy regularized off-policy RL algorithm, as our base RL algorithm. This regularization allows for a trade-off between the entropy of the policy and its expected return. Intuitively, increasing the entropy makes the policy more exploratory, which helps prevent convergence to poor solutions.

SAC learns a parameterized Q-function $Q_\theta(s, a)$ and policy $\pi_\phi(a|s)$, and an entropy-regularization term α. Q_θ is learned by minimizing a bootstrapped estimate of the Q-value using a target Q-value network with an included entropy term. π_ϕ is learned by minimizing the expected KL-divergence between $\phi_\phi(\cdot|s_t)$ and $\frac{\exp\{Q_\theta(s_t,\cdot)\}}{Z_\theta(s_t)}$, where $Z_\theta(s_t)$ is a normalization constant. Lastly, the entropy regularization term α is learned by iteratively adjusting to fit a target entropy. We choose to use SAC over existing off-policy methods since it has shown consistently better results in many popular domains.

IV. Approach

We now describe our learning framework for efficient deformable object manipulation. We start by the pick and place problem. Following this, we discuss our algorithm.

A. Deformable Object Manipulation as a Pick and Place Problem

We look at a more amenable action space while retaining the expressivity of the general action space: pick and place. The pick and place action space has had a rich history in planning with rigid objects [4, 30]. Here, the action space is the location to pick (or grasp) the object a^t_{pick} and the location to place (or drop) the object a^t_{place}. This operation is done at every step t, but we will drop the superscript for ease of reading. With rigid objects, the whole object hence moves according $a_{pick} \to a_{place}$. However, for a deformable object, only the point corresponding to a_{pick} on the object moves to a_{place}, while the other points move according to the kinematics and

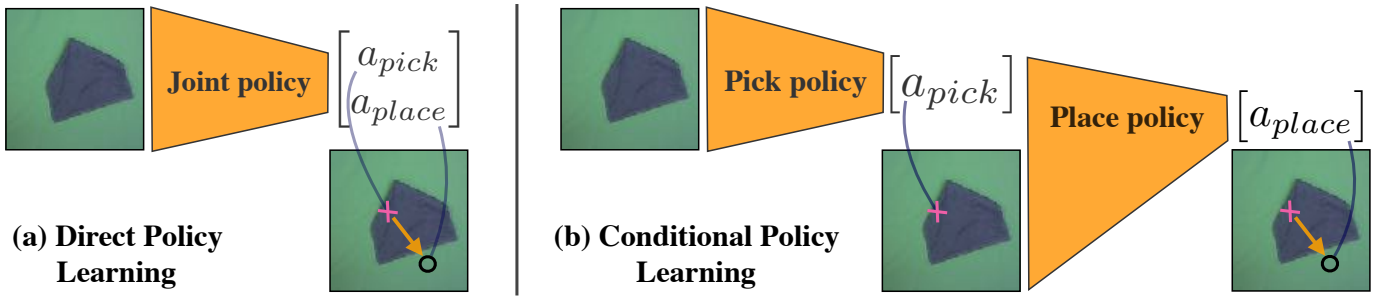

Fig. 2: In direct policy learning (a), the policy directly outputs both the pick and the place location. While in conditional policy learning, the composite action space is broken down into a separate picking and placing policy, where the placing policy takes the output of the picking policy as input.

dynamics of the deformable object [41]. Empirically, since in each action the robot picks and places a part of the deformable object, there is significant motion in the object, which means that the robot gets a more informative reward signal after each action. Also note that this setting allows for multiple pick-and-place operations that are necessary for tasks such as spreading out a scrunched up piece of cloth.

B. Learning with Composite Action Spaces

The straightforward approach to learning with a pick-place action space is to learn a policy π_{joint} that directly outputs the optimal locations to pick and to place $[a_{pick}, a_{place}]$, i.e. $\pi_{joint} \equiv p(a_{pick}|o) \cdot p(a_{place}|o)$ where o is the observation of the deformable object (Fig. 2(a)). However, this approach fails to capture the underlying composite and conditional nature of the action space, where the location to place a_{place} is strongly dependent on the pick point a_{pick}.

One way to learn with conditional output spaces is to explicitly factor the output space during learning. This has provided benefits in several other learning problems from generating images [64] to predicting large dimensional robotic actions [40, 62]. Hence instead of learning the joint policy, we factor the policy as:

$$\pi_{factor} \equiv \pi_{pick}(a_{pick}|o) \cdot \pi_{place}(a_{place}|o, a_{pick}) \quad (1)$$

This factorization will allow the policy to reason about the conditional dependence of placing on picking (Fig. 2(b)). However, in the context of RL, we face another challenge: action credit assignment. Using RL, the reward for a specific behavior comes through the cumulative discounted reward at the end of an episode. This results in the *temporal credit assignment* problem where attributing the reward to a specific action is difficult. With our factored action spaces, we now have an additional credit assignment problem on the different factors of the action space. This means that if an action receives high reward, we do not know if it is due to π_{pick} or π_{place}. Due to this, training π_{factor} jointly is inefficient and often leads to the policy selecting a suboptimal pick location. This suboptimal π_{pick} then does not allow π_{place} to learn, since $\pi_{place}(a_{place}|o, a_{pick})$ only sees suboptimal picking locations a_{pick} during early parts of training. Thus, this leads to a mode collapse as shown in Sec. V-D.

To overcome the action credit assignment problem, we propose a two-stage learning scheme. Here the key insight is that training a placing policy can be done given a full-support picking policy and the picking policy can be obtained from the placing policy by accessing the Value approximator for placing. Algorithmically, this is done by first training π_{place} conditioned on picking actions from the uniform random distribution $\mathbf{U}_{pick}$. Using SAC, we train and obtain $\pi_{place}(a_{place}|o, a_{pick})$, s.t. $a_{pick} \sim \mathbf{U}_{pick}$ as well as the place value approximator $V^{\pi_{place}}_{place}(o, a_{pick})$. Since the value is also conditioned on pick point a_{pick}, we can use this to obtain our picking policy as:

$$\pi_{pick} \equiv \arg\max_{a_{pick}} V^{\pi_{place}}_{place}(o, a_{pick}) \quad (2)$$

We call this picking policy: Maximum Value under Placing (MVP). The $\arg\max$ is computed by searching over all available pick location from the image of the object being manipulated. MVP allows us get an informed picking policy without having to explicitly train for picking. This makes training efficient for off-policy learning with conditional action spaces especially in the context of deformable object manipulation.

V. Experimental Evaluation

In this section we analyze our method MVP across a suite of simulations and then demonstrate real-world deformable object manipulation using our learned policies.

A. Cloth Manipulation in Simulation

Most current RL environments like OpenAI Gym [3] and DM Control [61], offer a variety of rigid body manipulation tasks. However, they do not have environments for deformable objects. Therefore, for consistent analysis, we build our own simulated environments for deformable objects using the DM Control API. To simulate deformable objects, we use composite objects from MuJoCo 2.0 [63]. This allows us to create and render complex deformable objects like cloths and ropes. Using MVP, we train policies both on state (locations of the composite objects) and image observations ($64 \times 64 \times 3$

Fig. 3: Learning comparisons between baselines and our method on the three deformable object manipulation environments with state-based training in simulation. The dotted black line is computed by evaluating MVP on the final learned 'learned placing with uniform pick' policy. Each experiment was run on 4 random seeds.

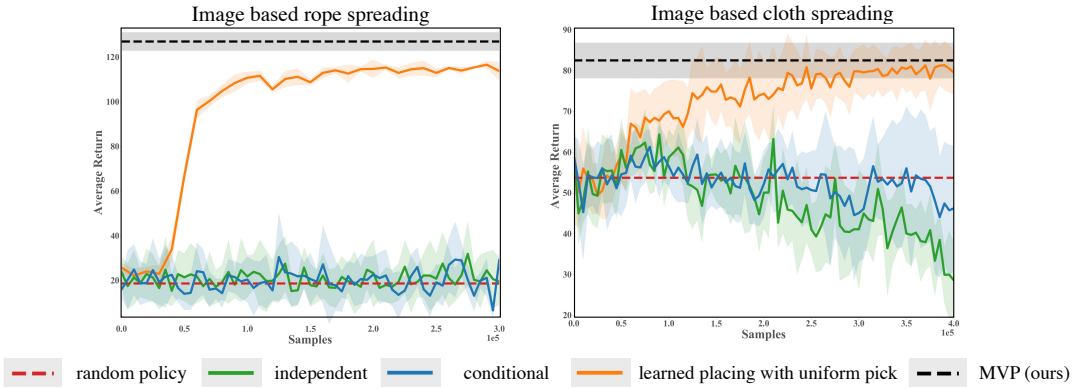

Fig. 4: Learning comparisons between baselines and our method on two deformable object manipulation environments with image-based training in simulation. Note that we do not include the *cloth-simplified* environment here since image-based transfer to real robot would involve corner detection. The dotted black line is computed by evaluating MVP on the final learned 'learned placing with uniform pick' policy. Each experiment was run on 3 random seeds.

RGB). For image-based experiments, we uniformly randomly select a pick point on a binary segmentation of the cloth or rope in order to guarantee a pick point on the corresponding object. Images are segmented using simple color channel thresholding. The details for the three environments we use are as follows:

1. Rope : The goal is to stretch the rope (simulated as a 25 joint composite) horizontally straight in the center of the table. The action space is divided into two parts as a_{pick} and a_{place}. a_{pick} is the two dimension pick point on the rope, and a_{place} is the relative distance to move and place the rope. All other parts of the rope move based on the simulator dynamics after each action is applied. We constrain the relative distance to move in a small radius around the pick point due to unstable simulations for larger movements. The reward for this task is computed from the segmentation of the rope in RGB images as:

$$reward = \sum_{i=1}^{H} e^{0.5\times|i-32|} \sum_{j=1}^{W} s_{i,j}, \tag{3}$$

where i is the row number of the image, j is the column number, $s_{i,j}$ is the binary segmentation at pixel location (i, j), and W, H correspond to height and width. Hence for a 64×64 image the reward encourages the rope to be in the center row (row number 32) with an exponential penalty on rows further from the center. At the start of each episode, the rope is initialized by applying a random action for the first 50 timesteps.

2. Cloth-Simplified : The cloth consists of an 81 joint composite that is a 9×9 grid. The robot needs to pick the corner joint of the cloth and move that to the target place. The action space is similar to the rope environment except the picking location can only be one of the four corners. In this environment, the goal is to flatten the cloth in the middle of the table. Our reward function is the intersection of the binary mask of the cloth with the goal cloth configuration.

3. Cloth : In contrast to the *Cloth-Simplified* environment that can only pick one of the 4 corners, *Cloth* allows picking any point in the pixel of cloth (if it is trained with image observation) or any composite particle (if state observation is used). The reward used is the same as in *Cloth-Simplified*.

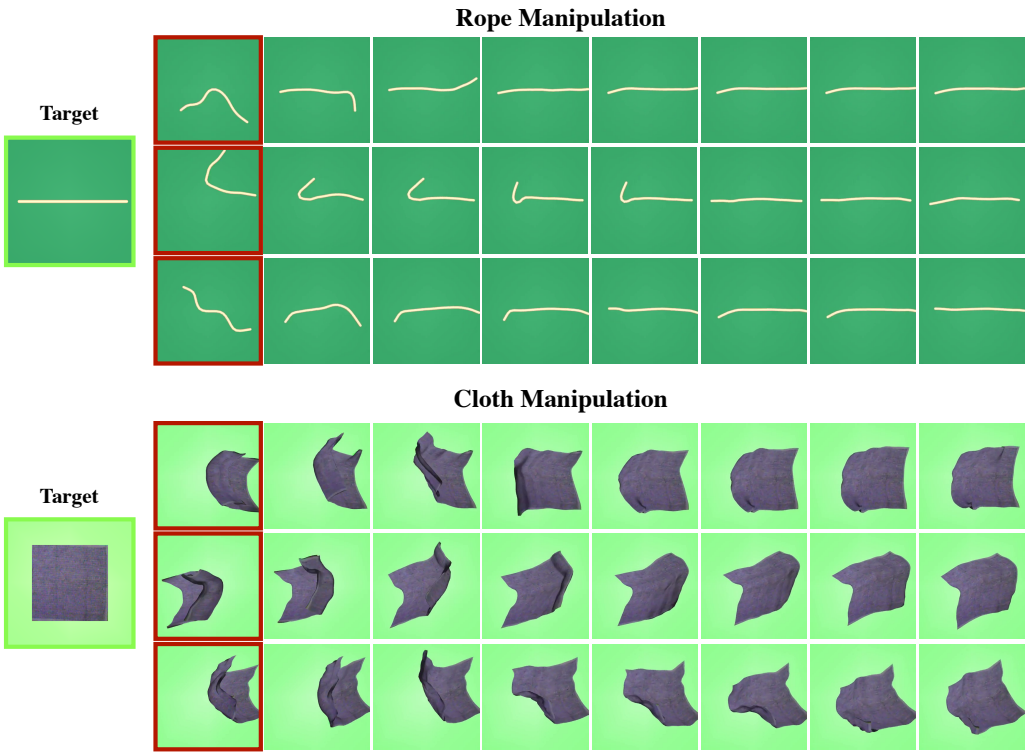

Fig. 5: We demonstrate deformable object manipulation in the simulated environments using our learned MVP policy. In the top half, we see the policy successfully horizontally straightens and centers a rope in the top. And in the bottom half, we see our method successfully spreading out a cloth from multiple starting states. Each image is about 5 actions apart for rope experiments, and 10 actions for cloth experiments.

For both the Cloth and Cloth-Simplified environments, the cloth is initialized by applying a random action for the first 130 timesteps of each episode. In MuJoCo, the skin of the cloth can be simulated by uploading an image taken of a real cloth texture. In the domain randomization experiments, we randomize the cloth by switching out different textures.

B. Learning Methods for Comparison

To understand the significance of our algorithm, we compare the following learning methods: random, independent, conditional, learned placing with uniform pick, and MVP (ours) as described below.

- **Random**: We sample pick actions uniformly over available pick locations and place actions uniformly over the action space of the robot.
- **Independent / Joint**: We use a joint factorization of $p(a_{pick}, a_{place}|o)$ by simultaneously outputting the a_{pick} and a_{place}. Alternatively, we label it as Independent to distinguish it from the Conditional baseline.
- **Conditional**: We first choose a pick location, and then choose a place vector distance given the pick location, modeled as $p(a_{pick}|o) \times p(a_{place}|a_{pick}, o)$.
- **Learned Placing with Uniform Pick**: We use the conditional distribution $p(a_{place}|a_{pick}, o)$, where a_{pick} is uniformly sampled from available pick locations.
- **MVP (ours)**: We use the trained learned placing with uniform pick policy and choose a_{pick} by maximizing over the learned Q-function.

Our experimental results for various model architectures on the rope and cloth environments are shown in Fig. 3 and Fig. 4. We trained the Independent and Conditional baselines using a modified environment, where an extra positive reward is given for successfully outputting a pick location on the cloth or rope. This is to allow a more fair comparison with the Random, Learned Placing with Uniform Pick, and MVP (ours) policies which have prior access to the image segmentations.

C. Training Details

For the training in the simulation, we use SAC [14] as our off-policy algorithm and make a few modifications on the rlpyt code-base [58]. For state-based experiments, we use an MLP with 2 hidden layers of 256 units each; approximately 150k parameters. For image-based experiments, we use a CNN with 3 convolutional layers with channel sizes 64, 64, and 4, accordingly, and each with a kernel size of 3 and a stride of 2. This is followed by with 2 fully connected hidden layers of 256 units each. In total approximately 200k parameters are learned. For all models, we repeat the pick information 50 times before concatenating with the state observations or flattened image embeddings so that the pick information and the observation embeddings are weighted equally, which improves performance. The horizon for Rope is 200 and 120

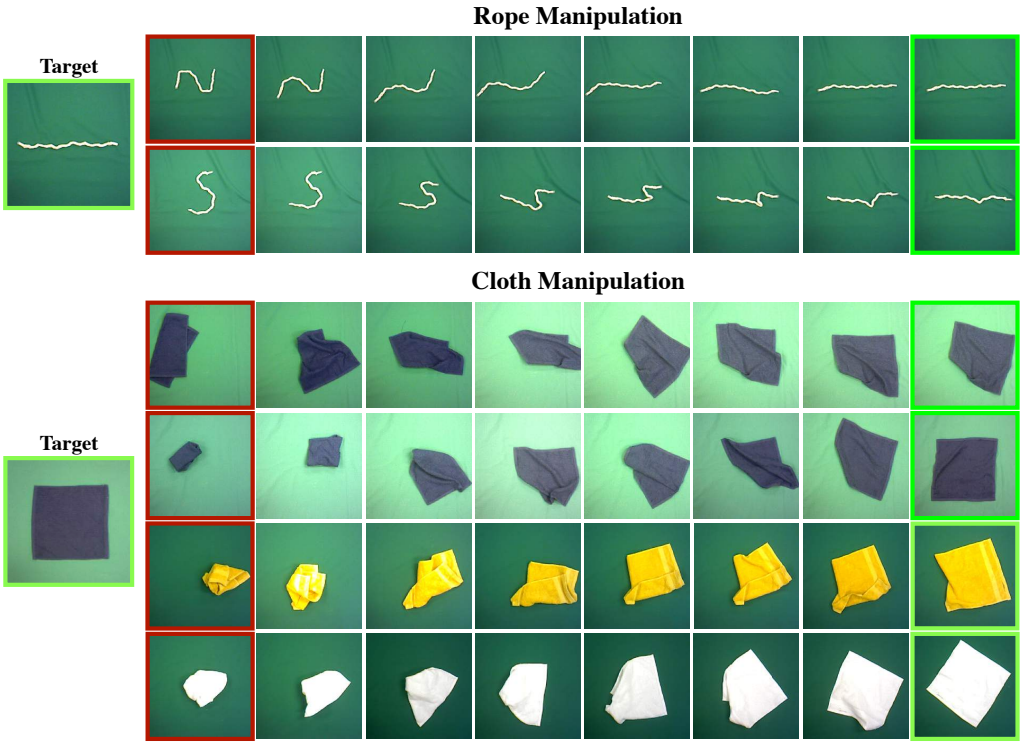

Fig. 6: Using MVP for learning the policy along with domain randomization for simulation to real transfer, we demonstrate deformable object manipulation on a real robot for a rope (top) and a cloth (bottom). In both examples, the task is to spread out the object to reach the target spread out configuration in the middle of the table (left) for two different start locations (in red). For rope spreading, each frame corresponds to one pick-place action taken by our PR2 robot (Fig. 1(a)), while for cloth spreading each frame corresponds to 10 actions on our robot.

for both Cloth environments. The minimum replay pool size is 2000 for Rope and 1200 for the Cloth environments. The image size used for all environments is $64 \times 64 \times 3$. We perform parallel environment sampling to speed-up overall training by $3-5$ times. For both rope and cloth experiments, the total compute time on one TitanX GPU and 4 CPU cores is roughly 4-6 hours. In the case of rope experiments, a reasonable policy can be obtained in one sixth of the training time, and about a half of the training time for cloth experiments. All of our training code, baselines, and simulation environments will be publicly released.

D. Does conditional pick-place learning help?

To understand the effects of our learning technique, we compare our learned placing with uniform pick technique with the independent representation in Fig. 3. We can see that using our proposed method shows improvement in learning speed for state-based cloth experiments, and image-based experiments in general. The state-based rope experiments do not show much of a difference due to the inherent simplicity of the tasks. Our method shows significantly higher rewards in the cloth simplified environment, and learns about 2X faster in the harder cloth environment. We also note that the independent and conditional baselines perform better on the full state-based cloth environment compared to when constraining the task to four corner pick points (Cloth-Simplified). This most likely occurs since the Cloth-Simplifed task structures its pick action as 4 discrete locations, which increases the likelihood of mode collapse on a single corner compared to when using a continuous pick representation for the full Cloth environment. For image-based experiments, the baseline methods do no better than random while our method gives an order of magnitude (5-10X) higher performance for reward reached. The independent and conditional factored policies for image-based cloth spreading end up performing worse than random, suggesting mode collapse that commonly occurs in difficult optimization problems [11]. Note that to strengthen the baselines, we add additional rewards to bias the pick points on the cloth; However, this still does not significantly improve performance for the challenging image based tasks. This demonstrates that conditional learning indeed speeds up learning for deformable object manipulation especially when the observation is an image.

E. Does setting the picking policy based on MVP help?

One of the key contributions of this work is to use the placing value to inform the picking policy (Eq. 2) without explicitly training the picking policy. As we see in both state-based (Fig. 3) and image-based case (Fig. 4) training with MVP gives consistently better performance. Even when our conditional policies with uniform pick location fall below the baselines as seen in Cloth (State) and Rope (State),

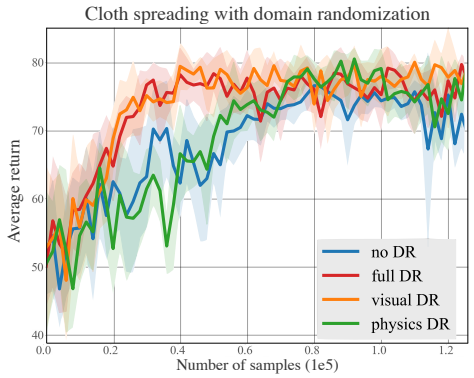

Fig. 7: Learning comparisons between different forms of domain randomization (DR) on cloth-spreading trained with MVP. This is evaluated in simulation across 5 random seeds and shaded with $\pm$ 1 standard deviation.

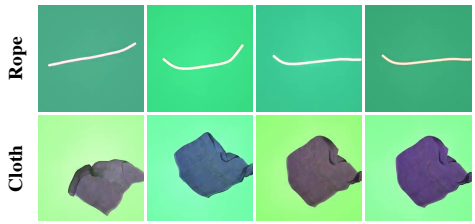

Fig. 8: Examples of domain randomization applied in the rope and cloth environments.

using MVP significantly improves the performance. Note that although MVP brings relatively smaller boosts in performance compared to the gains brought by the learned placing with uniform pick method, we observe that the learned placing with uniform pick policy already achieves a high success rate on completing the task, and even a small boost in performance is visually substantial when running evaluations in simulation and on our real robot.

F. How do we transfer our policies to a real robot?

To transfer our policies to the real-robot, we use domain randomization (DR) [62, 45, 48] in the simulator along with using images of real cloths. Randomization is performed on visual parameters (lighting and textures) as well physics (mass and joint friction) of the cloth. Examples of randomized observations can be seen in Fig. 8. Additionally, in simulation evaluation, we notice no degradation in performance due to DR while training using MVP as seen in Fig. 7. Physics randomization most likely had little-to-no benefit (compared to visual randomization) to the learning process due to the fact that the simulator itself is already a little noisy.

In order to perform actions on our PR2 robot, we first calibrate pixel-space actions with robot actions. This is done by collecting 4-5 points mapping between robot x, y coordinates to image row, column pixel locations, and fitting a simple linear map. Next, we capture RGB images from a head-mounted camera on our PR2 robot(Fig. 1(a)) and input the image into our policy learned in the simulator. Since a_{pick} and a_{place} are both defined as points on the image, we can easily command the robot to perform pick-place operations on the deformable object placed on the green table by planning with Moveit! [5].

G. Evaluation on the real robot

We evaluate our policy on the *rope-spread* and *cloth-spread* experiments. As seen in Fig. 6, policies trained using MVP are successfully able to complete both spreading tasks. For our cloth spreading experiment, we also note that due to domain randomization, a single policy can spread cloths of different colors. For quantitative evaluations, we select 4 start configurations for the cloth and the rope and compare with various baselines (Table I) on the spread coverage metric. For the rope task, we run the policies for 20 steps, while for the much harder cloth task we run policies for 150 steps. The large gap between MVP trained policies and independent policies supports our hypothesis that the conditional structure is crucial for learning deformable object manipulation. Robot execution videos can be accessed from the video submission. We observe that, compared to our simulation policy which solves the manipulation tasks in 20-30 actions, the robot sometimes makes unnecessary manipulation actions. This may be attributed to a combination of a sim-to-real gap, and deficiencies of the robot (e.g. the robot would miss its pick, or its thick gripper would pick up both layers of a folded cloth).

Domains	Random policy	Conditional Pick-Place	Independent / Joint policy	MVP (ours)
Rope	0.34	0.16	0.21	**0.48**
Cloth	0.59	0.34	0.32	**0.84**

TABLE I: Average goal area intersection coverage for rope and cloth spreading tasks on the PR2 robot.

VI. Conclusion and Future Work

We have proposed a conditional learning approach for learning to manipulating deformable objects. We have shown this significantly improves sample complexity. To our knowledge, this is the first work that trains RL from scratch for deformable object manipulation and demonstrates it on real robot. We believe this work can open up many exciting avenues for deformable object manipulation from bubble wrapping a rigid object to folding a T-shirt, which pose additional challenges in specifying a reward function and handling partial observability. Additionally, since our technique only assumes an actor-critic algorithm, we believe it can be combined with existing learning from demonstration based techniques to obtain further improvements in performance.

VII. Acknowledgements

We thank AWS for computing resources and Boren Tsai for support in setting up the robot. We also gratefully acknowledge the support from Komatsu Ltd., The Open Philanthropy Project, Berkeley DeepDrive, NSF, and the ONR Pecase award.

References

[1] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[2] Dmitry Berenson. Manipulation of deformable objects without modeling and simulating deformation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4525–4532. IEEE, 2013.

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

[4] Rodney A Brooks. Planning collision-free motions for pick-and-place operations. *The International Journal of Robotics Research*, 2(4):19–44, 1983.

[5] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.

[6] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[7] Nabil Essahbi, Belhassen Chedli Bouzgarrou, and Grigore Gogu. Soft material modeling for robotic manipulation. In *Applied Mechanics and Materials*, volume 162, pages 184–193. Trans Tech Publ, 2012.

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

[9] Barbara Frank, Cyrill Stachniss, Nichola Abdo, and Wolfram Burgard. Efficient motion planning for manipulation robots in environments with deformable objects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2180–2185. IEEE, 2011.

[10] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv e-prints*, art. arXiv:1802.09477, Feb 2018.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[12] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In *Advances in Neural Information Processing Systems*, pages 9094–9104, 2018.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv e-prints*, art. arXiv:1801.01290, Jan 2018.

[14] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[15] Dominik Henrich and Heinz Wörn. *Robot manipulation of deformable objects*. Springer Science & Business Media, 2012.

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

[17] Zhe Hu, Peigen Sun, and Jia Pan. Three-dimensional deformable object manipulation using fast online gaussian process regression. *IEEE Robotics and Automation Letters*, 3(2):979–986, 2018.

[18] Biao Jia, Zhe Hu, Zherong Pan, Dinesh Manocha, and Jia Pan. Learning-based feedback controller for deformable object manipulation. *arXiv preprint arXiv:1806.09618*, 2018.

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

[20] Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, et al. Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015.

[21] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[22] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[23] Fouad F Khalil and Pierre Payeur. Dexterous robotic manipulation of deformable objects with multi-sensory feedback-a review. In *Robot Manipulators Trends and Development*. IntechOpen, 2010.

[24] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[25] Alex X Lee, Abhishek Gupta, Henry Lu, Sergey Levine, and Pieter Abbeel. Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5265–5272. IEEE, 2015.

[26] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

[27] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *ISER*, 2016.

[28] Timothy P. Lillicrap, Jonathan J. Hunt, Alexand er Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David

Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv e-prints arXiv:1509.02971*, 2015.

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

[30] Tomás Lozano-Pérez, Joseph L. Jones, Emmanuel Mazer, and Patrick A. O'Donnell. Task-level planning of pick-and-place robot motions. *Computer*, 22(3):21–29, 1989.

[31] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *ICRA*, 2016.

[32] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315. IEEE, 2010.

[33] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:1806.07851*, 2018.

[34] Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jürgen Schmidhuber. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14):1521–1537, 2008.

[35] Dale McConachie and Dmitry Berenson. Estimating model utility for deformable object manipulation using multiarmed bandit methods. *IEEE Transactions on Automation Science and Engineering*, 15(3):967–979, 2018.

[36] Dale McConachie, Mengyao Ruan, and Dmitry Berenson. Interleaving planning and control for deformable object manipulation. In *International Symposium on Robotics Research (ISRR)*, 2017.

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

[38] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1602.01783, Feb 2016.

[39] Mark Moll and Lydia E Kavraki. Path planning for deformable linear objects. *IEEE Transactions on Robotics*, 22(4):625–636, 2006.

[40] Adithyavairavan Murali, Lerrel Pinto, Dhiraj Gandhi, and Abhinav Gupta. Cassl: Curriculum accelerated self-supervised learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6453–6460. IEEE, 2018.

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

[42] David Navarro-Alarcon, Yun-hui Liu, Jose Guadalupe Romero, and Peng Li. On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments. *The International Journal of Robotics Research*, 33(11):1462–1480, 2014.

[43] Piotr Pierański, Sylwester Przybył, and Andrzej Stasiak. Tight open knots. *The European Physical Journal E*, 6 (2):123–128, 2001.

[44] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *ICRA*, 2016.

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

[46] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[47] Samuel Rodriguez, Xinyu Tang, Jyh-Ming Lien, and Nancy M Amato. An obstacle-based rapidly-exploring random tree. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 895–900. IEEE, 2006.

[48] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[49] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Transactions on Robotics*, 23(6):1141–1150, 2007.

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

[51] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. Generalization in robotic manipulation through the use of non-rigid registration. In *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*, 2013.

[52] John Schulman, Alex Lee, Jonathan Ho, and Pieter Abbeel. Tracking deformable objects with point clouds. In *2013 IEEE International Conference on Robotics and Automation*, pages 1130–1137. IEEE, 2013.

[53] John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.

[54] 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. *arXiv preprint arXiv:1809.09810*, 2018.

[55] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. Deep imitation learning of sequential fabric smoothing policies. *arXiv preprint arXiv:1910.04854*, 2019.

[56] Karun B Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996.

[57] Jerzy Smolen and Alexandru Patriciu. Deformation planning for robotic soft tissue manipulation. In *2009 Second International Conferences on Advances in Computer-Human Interactions*, pages 199–204. IEEE, 2009.

[58] Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

[59] Jan Stria, Daniel Prusa, Vaclav Hlavac, Libor Wagner, Vladimir Petrik, Pavel Krsek, and Vladimir Smutny. Garment perception and its folding using a dual-arm robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 61–67. IEEE, 2014.

[60] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

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

[62] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018.

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

[64] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

[65] Takahiro Wada, Shinichi Hirai, Sadao Kawamura, and Norimasa Kamiji. Robust manipulation of deformable objects by a simple pid feedback. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 85–90. IEEE, 2001.

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

[67] Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. Learning robotic manipulation through visual planning and acting. *arXiv preprint arXiv:1905.04411*, 2019.

[68] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[69] Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics—a review. *Sensors and Actuators A: physical*, 167(2):171–187, 2011.