FlingBot: The Unreasonable Effectiveness of Dynamic Manipulation for Cloth Unfolding

Anonymous Author(s)
Affiliation
Address
email

Abstract: High-velocity dynamic actions (e.g., fling or throw) play a crucial role in our everyday interaction with deformable objects by improving our efficiency and effectively expanding our physical reach range. Yet, most prior works have tackled cloth manipulation using exclusively single-arm quasi-static actions, which requires a large number of interactions for challenging initial cloth configurations and strictly limits the maximum cloth size by the robot's reach range. In this work, we demonstrate the effectiveness of dynamic flinging actions for cloth unfolding with our proposed self-supervised learning framework, FlingBot. Our approach learns how to unfold a piece of fabric from arbitrary initial configurations using a pick, stretch, and fling primitive for a dual-arm setup from visual observations. The final system achieves over 80% coverage within 3 actions on novel cloths, can unfold cloths larger than the system's reach range, and generalizes to T-shirts despite being trained on only rectangular cloths. We also finetuned FlingBot on a real-world dual-arm robot platform, where it increased the cloth coverage over 4 times more than the quasi-static baseline did. The simplicity of FlingBot combined with its superior performance over quasi-static baselines demonstrates the effectiveness of dynamic actions for deformable object manipulation. Code, data, and simulation environment will be publicly available.

Keywords: Dynamic manipulation, Cloth manipulation, Self-supervised learning

1 Introduction

High-velocity dynamic actions play a crucial role in our everyday interaction with deformable objects. Making our beds in the morning is not effectively accomplished by picking up each corner of the blanket and placing them in the corresponding corners of the bed, one by one. Instead, we are more likely to grasp the blanket with two hands, stretch it, and then unfurl it with a fling over the bed. This fluid high-velocity flinging action is an example of dynamic manipulation [1], which is used to improve our physical reachability and action efficiency – allowing us to unfold large crumpled cloths with as little as one interaction.

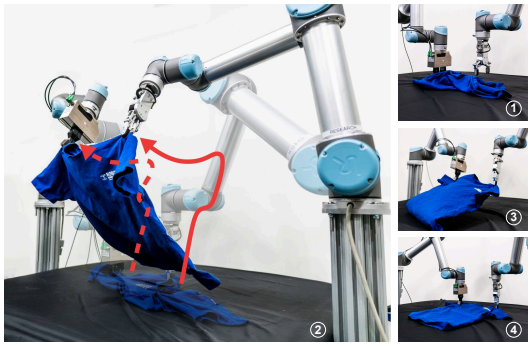

Figure 1: **Cloth unfolding with dynamic interactions.** Given a severely crumpled cloth, FlingBot uses a high-speed fling to unfurl the cloth with as little as one interaction. In this paper, we demonstrate that such dynamic actions can efficiently unfold cloths, generalizable to different cloth types, and improve the effective reach range of the system.

From goal-conditioned folding [2] to fabric smoothing [3, 4], prior works have achieved success using exclusively single-arm quasi-static interactions (e.g., pick & place) for cloth manipulation. However, these approaches require a large number of interactions for challenging initial configurations (e.g., highly crumpled cloths) or rely on strong assumptions about the cloth (e.g., predefined keypoints). Additionally, since the robot arm cannot manipulate the cloth at locations it can't reach, the maximum cloth size is greatly limited by the robot arm's reach range.

In this work, we focus on the task of cloth unfolding, a typical first step for many cloth manipulation tasks. The goal of this task is to maximize the cloth's coverage through interactions to expose the key visual features for downstream perception and manipulation. Therefore, an ideal cloth unfolding approach should be:

- **Efficient**: the approach should reach a high coverage with a small number of actions from arbitrarily crumpled initial configurations.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

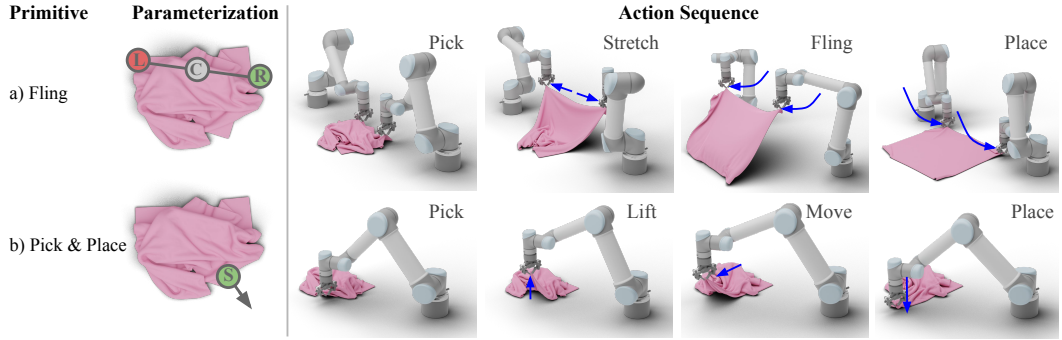

Figure 2: **Action Primitives.** The dynamic **Fling** primitive starts with a two-arm grasp at the left L and right R grasp locations with center point C, followed by a fixed stretch, fling, and place routine. A good fling could unfold a highly crumpled cloth in as little as a single step. In contrast, the quasi-static **Pick & Place** primitive, which grasps at the start location S, lifts, moves, then places at the end position specified by the arrow tip, requires many steps for such challenging cloth configurations. The quasi-static **Pick & Drag** primitive is just a Pick & Place without the lift step.

- **Generalizable**: the algorithm should not rely on heuristics (e.g., grasp predefined key points, etc.). This is especially important when unfolding is only the initial stage of a cloth perception and manipulation pipeline, where key points are not visible or severely occluded, and when the system must handle cloth types unseen during training, which may not contain the predefined key points.
- **Flexible Beyond the Workspace**: the approach should work with cloths of different sizes, including large ones which fall outside the robot's physical reach range.

To achieve this goal, we present FlingBot, a self-supervised algorithm that learns how to unfold a piece of fabric from arbitrary initial configurations using a pick, stretch, and fling primitive for a dual-arm setup. At each time step, the policy predicts value maps from its visual observation and picks actions greedily with respect to its value maps. To provide the supervision signal, the system computes the difference in coverage of the cloth before and after each action – the delta-coverage – from the visual input captured by a top-down camera. FlingBot achieves over 80% coverage within 3 actions on novel cloths and increases the cloth's coverage by more than twice that of pick & place and pick & drag quasi-static baselines on rectangular cloths. Our approach is flexible to large cloths whose dimensions exceed the robot arm's reach ranges and generalizes to T-shirts despite being trained on rectangular cloths. We fine-tune our approach in the real world, where, on average, it increased the cloth coverage over 4 times more than the quasi-static pick & place baseline did.

In summary, our contributions are three-fold:

- Our main contribution is in demonstrating the effectiveness of dynamic manipulation for cloth unfolding through our self-supervised learning framework, FlingBot.
- We propose a parameterization for the dual-arm grasp of our fling primitive, which enables the application of the simple yet effective single-arm grasping technique spatial action maps [5, 6, 2, 7] to dual-arm grasping while satisfying safety constraints of dual-arm systems. The simplicity of FlingBot combined with its superior performance over quasi-static baselines further emphasize the effectiveness of dynamic actions for deformable object manipulation.
- We present a custom simulator built on top of PyFlex [8, 9], a particle-based CUDA accelerated simulator, which supports the loading of arbitrarily shaped cloth meshes. We hope this open-source simulator expands cloth manipulation research to more complex cloth types.

2 Related Work

A convincing argument for the *addition* of dynamic actions to previously exclusively quasi-static cloth manipulation pipelines would need to demonstrate their superior performance on a core cloth manipulation skill. As a typical first step for many cloth manipulation tasks, cloth unfolding is a popular and important problem setting for studying deformable object manipulation. The goal of cloth unfolding is to maximize the coverage of the cloth on the workspace, which exposes key visual features of the cloth for downstream applications. However, achieving a fully unfolded cloth configuration from a severely crumpled initial configuration remains a challenging problem. Additionally, doing so efficiently for many different types of cloths, some larger than the system's reach range, is extremely challenging.

Quasi-static Cloth Manipulation with Expert Demonstrations. Prior works have explored using heuristics and identifying key points such as wrinkles [10], corners [11, 12, 13], edges [14], or combinations of them [15, 15], but fail to generalize to severely self-occluded cloth configurations, to when required key

points aren't visible, or to non-square cloths. Recent works which use reinforcement learning have relied on expert demonstrations to learn cloth unfolding [3, 16] in quasi-static pick-and-drag action spaces. Although such approaches bypass the sample-inefficiency of the exploration problem, expert demonstrations can be sub-optimal and brittle if from hard-coded heuristics [3] or expensive if from humans [16].

Self-supervised Quasi-static Cloth Manipulation. Bypassing the dependency on an expert, self-supervised cloth manipulation has been demonstrated in unfolding with a factorized pick & place action space by Wu et al. [4] and goal conditioned folding using spatial action maps [7, 6, 5] by Lee et al. [2]. However, all these approaches operate entirely in quasi-static action spaces.

Dynamic Cloth Manipulation. In contrast to quasi-static manipulation, which are "operations that can be analyzed using kinematics, static, and quasi-static forces (such as frictional forces at sliding contacts)" [1], dynamic manipulation involves operations whose analysis additionally requires "forces of acceleration". Intuitively, dynamic manipulation (e.g., tossing [6]) involve high-velocity actions which build up objects' momentum, such that the manipulated objects continue to move after the robot's end-effector stops. Such dynamic manipulation results in an effective increase in reach range and a reduction in the number of actions to complete the task compared to exclusively quasi-static manipulation. However, prior works on dynamic cloth manipulation have either relied on cloth's vertex positions [17] (so is only practical in simulation), a motion capturing system with markers combined with human demonstrations [18], or custom hardware [19, 20].

While prior cloth unfolding works report high average final coverages from relative high average initial coverages, we emphasize that we consider severely crumpled initial cloth configurations, which are much more challenging yet still realistic (see section 4.2). Additionally, no prior works have considered unfolding cloths larger than the robot's reach range (indeed, Ganapathi et al. [16]'s workspace is significantly larger than their maximum cloth size due to physical reachability limitations). Finally, our algorithm learns through self-supervised trial and error directly from visual input without requiring expert demonstrations or ground truth state information.

3 Method

The goal of cloth unfolding is to manipulate a cloth from an arbitrarily crumpled initial state to a flattened state. Concretely, this amounts to maximizing the coverage of the cloth on the workspace surface. Intuitively, dynamic actions have the potential to achieve high performance on cloth unfolding by appropriately making use of the cloth's mass in a high-velocity action to unfold the cloth (Sec. 3.1).

From a top-down RGB image of the workspace with the cloth, our policy decides the next fling action (Sec. 3.2) by picking the highest value action which satisfies the system's constraints (Sec. 3.3). It predicts the value of each action with a value network (Sec. 3.4) which is trained in a self-supervised manner to take actions that maximally increases the cloth's coverage. The supervision signal is computed directly from the visual observation captured by the top-down camera. After training in simulation, we finetune and evaluate the model in the real world (Sec. 3.5).

3.1 Advantages of the Fling Action Primitive

Quasi-static actions, such as pick & place, rely on strategically chosen pick and place points and friction between the cloth and ground to stretch the cloth out. The complexity of friction forces between the workspace and the (nonvisible, ground-facing) surfaces of the cloth when being dragged means the cloth's final configuration may be difficult to predict from only visual observations, which may be reflected in the complexity of quasi-static unfolding policies. In addition, such systems can't manipulate points on the cloth to regions outside of their physical reach range, which limits their maximum cloth dimensions. Meanwhile, dynamic actions largely rely on cloths' mass combined with a high-velocity throw to do most of its work. Specifically, a single dynamic motion primitive can be effectively reused for most instances, so such systems can afford to learn a simpler grasping policy which generalizes better to different cloth types. In addition, they can reach higher coverages in smaller numbers of interactions and throw corners of very large cloths beyond the system's reach range, effectively expanding the reach range of the system.

We propose to use a dual-arm pick, stretch, and fling primitive. Here, a dual-arm system stretches the cloth between the arms, which unfolds it in one direction, and then flings the cloth, which makes use of the cloth's mass to unfold it in the other direction. The combination of stretching and flinging ensures that, given two appropriate grasp points, our motion primitive should be sufficient for single step unfolding when possible.

3.2 Fling Action Primitive Definition

To achieve an efficient, generalizable, and flexible cloth unfolding system, we argue that the system requires two arms operating in a dynamic action space. To this end, we've designed a pick, stretch, and fling motion primitive for a dual-arm system, where each arm is placed on either side of the cloth workspace, as follows.

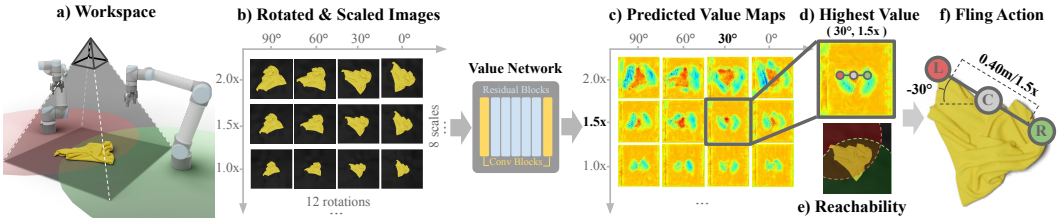

Figure 3: **Method Overview.** From a top-down RGB image a), our policy evaluates a batch of different action rotations and scales by transforming the observation b) then predicting the corresponding batch of value maps c). The highest value action d), corresponding to the maximum value pixel which also satisfies the arms' reachability constraints e) is chosen. Finally, the chosen pixel's location and its observation's transformation is decoded into the fling action parameters (i.e., center point, distance, relative orientation between gripper).

First, the arms perform a top-down pinch-grasp on the cloth at locations $L,R \in \mathbb{R}^3$. Second, the arms lift the cloth to 0.30m and stretch the cloth taut in between them. Third, the arms fling the cloth forward 0.70m at 1.2ms^{-1} then pull backwards 0.50m at 1.2ms^{-1}, which swings the cloth forwards. Finally, the arms place the cloth down to the workspace and release their grips. The entire motion primitive is visualized in (Fig. 2a).

Instead of parameterizing and learning all steps of our primitive, we can fix the stretching step to always stretch the cloth as much as possible without tearing the cloth. Further, a fixed flinging trajectory with a fixed flinging velocity can still be effective for typical cloth masses, sizes, and stiffnesses in our task datasets (see Sec. 4.2). To verify this claim, we also implemented a variant of FlingBot which also predicts the fling velocity (see Sec. 4.3). Thus, the problem of learning to pick, stretch, and fling reduces down to the problem of learning where to pick, so we parameterize our flinging primitive with only two grasp locations, one for each arm in the dual-arm system.

3.3 Constraint Satisfying Fling Action Parameterization

A dual-arm grasp is parameterized by two points $L,R \in \mathbb{R}^3$, which denotes where the left and right arm should approach from a top-down grasp respectively, and requires 6 scalars. Without loss of generality for the purposes of grasping from a top-down RGB-D input, the third dimension could be specified by depth information. This reduces L and R to two points in $\mathbb{R}^2$, each representing pixels to grasp from the visual input and uses only 4 scalars in total.

However, to minimize collisions between two arms, we wish to impose a constraint that L is always left of R, and vice versa. Additionally, the grasp width (i.e., the length of the line $L-R$) must not exceed the physical limit of the system and be smaller than the minimum safe distance limit between the two arms. Directly using L and R will entangle these two contraints (visualized in the supplementary material), making them difficult to satisfy. To make these constraints linear and independent, we propose an alternative 4-scalar parameterization, which consists of pixel position of the point $C \in \mathbb{R}^2$ at the center of the line $L-R$, an angle $\theta \in \mathbb{R}$ denoting the planar rotation of the line $L-R$, and a grasp width $w \in \mathbb{R}$ denoting the length of the line in pixel space. To constrain L to be on the left of R, we can constrain $\theta \in [-90°, 90°]$, while w can be directly constrained to appropriate system limits.

3.4 Learning Delta-Coverage Maximizing Fling Actions

The naive approach for learning the action parameters $\langle C_x, C_y, \theta, w \rangle$ by directly predicting these 4 scalars does not accommodate the best grasp's equivariances to the cloth's physical transformations. However, the learner should be equipped with the inductive bias that cloths in identical configurations have identical optimal grasp points for flinging relative to the cloth, regardless of the cloth's translation, rotation, and scale.

To this end, we propose to use spatial action maps [5, 6, 7]. By predicting grasp values with constant scale and rotation *in pixel space* from the transformed images, spatial action maps can recover grasp values with varying scales and rotations *in world space* by varying the transformation applied to the image. Concretely, given a visual observation from the top-down view (Fig. 3a), we generate a batch of rotated and scaled observations (Fig. 3b) then predict the corresponding batch of dense value maps (Fig. 3c). Each pixel in each value map contains the value of the action parameterized by that pixel's location, giving C, and its observation's rotation and scaling, giving θ and w respectively (Fig. 3f). The value network is supervised to regress each pixel in the value map to the ground truth delta-coverage – the difference in coverage before and after the action. Thus, by picking the grasping action with the highest value (Fig. 3d), the system picks grasp points which it expects would lead to the greatest increase in cloth coverage. By its architecture, the value network is invariant to translation. By rotating and scaling its inputs, it is also invariant to rotation and scale.

In our approach, we use 12 rotations, which discretizes the range $[-90°, 90°]$, and 8 scale factors in the range $[1.00, 2.75]$ at 0.25 intervals. Our value network is a fully convolutional neural network with nine residual blocks [21] and two convolutional layers in the first and last layer, and takes as input 64×64 RGB images. To evaluate actions within a useful range of scales, we crop the image such that the cloth takes up at most two-thirds of the image width and height before shrinking it by the inverse of the 8 scale factors above. We also filter out and reject all grasp point pairs which are out of reach for either arm.

Self-Supervised Learning. The value network is trained end-to-end with self-supervised trials in simulation then finetuned in the real world. In each training episode, a random task is sampled. Each fling step is labeled with its unfolding performance (i.e., delta coverage). The delta coverage is normalized, and computed by counting cloth mask pixels from a top-down view, then dividing by the cloth mask pixel count of a flattened cloth state. While our experiments have plain black workspace surface colors and non-textured colored cloths for easy masking, our approach can be combined with image segmentation or background subtraction approach to obtain the corresponding cloth mask to work with arbitrary cloth and workspace textures. Note that this supervision signal (i.e., cloth mask) is only needed during training.

The policy interacts with the cloth until it reaches 10 timesteps or predicts grasps on the workspace. The latter stopping condition indicates that the policy does not expect any possible action to further improve the cloth coverage. All policies are trained in simulation until convergence, which takes around 90,000 simulation steps, or 4 days on a machine with a GTX 1080 Ti. The network is trained using the Adam optimizer with a learning rate of 1e-3 and a weight decay of 1e-6.

3.5 Experiment Setup

Simulation Setup. Our custom simulation environment is built on top of the PyFleX [8] bindings to Nvidia FleX provided by SoftGym [9]. On top of the functionality provided by PyFleX and SoftGym, our simulation can load arbitrary cloth meshes, such as T-shirts, through the Python API and will be released along with code and data. While our simulation environment does not support loading URDFs of robots, we found it sufficient to represent only the arms' end effectors, and apply appropriate physical constraints to their locations. Specifically, we precompute a reachability map for each arm using PyBullet [22], and check each proposed grasp location against the corresponding reachability map. The observations are rendered using Blender 3D where the cloth HSV color is sampled uniformly between [0.0,1.0], [0.0,1.0], and [0.5,1.0] respectively, and the background is dark grey with a procedurally generated normal map to mimic the wrinkles on the real-world workspace surface. We further apply brightness, contrast, and hue jittering on observations in the data loader to help with the transfer to real. To qualitatively match the real-world cloth dynamics, our flinging primitive in simulation flings with speed 0.60ms^{-1} instead of the real world fling speed 1.20ms^{-1}.

Real World Setup. Our real-world experiment setup consists of two UR5s, where one is equipped with a Schunk WSG50 and the other with an OnRobot RG2, facing each other and positioned 1.35m apart. The top-down RGB-D image is captured with an Intel RealSense D415. To help with pinch grasp success in real, we used a 2-inch rubber mat as the workspace and added sponge tips to the gripper fingers. To mitigate the reliance on specialized and expensive force sensor, we chose to implement stretched cloth detection using only a second Intel RealSense D415 capturing a frontal view of the workspace. By segmenting the cloth from the front camera RGBD view of the workspace, then checking whether the top edge of the cloth mask is flat as a proxy for the cloth being stretched, we can pull the arms' end effectors apart until the top of the cloth is no longer bent.

4 Evaluation

We design a series of experiments to evaluate the advantage of dynamic actions over quasi-static actions in the task of cloth unfolding. The system's performance is evaluated on its efficiency (i.e., reaching high coverages with a small number of actions), generalization to unseen cloth types (i.e., flattening different types of T-shirts when only trained on rectangle cloths), and reach range (i.e., on cloths with dimensions larger than its reach range). Finally, we evaluate the algorithm's performance on the real-world setup.

4.1 Metrics

The algorithm performance is measured by the final coverage, delta-coverage from initial coverage, and the number of interactions. Final coverage measures the coverage at the end of an episode, whereas delta-coverage is final coverage minus initial coverage. All our coverage statistics are normalized (i.e., divided by the maximum possible coverage of the cloth in a flattened configuration) and can be easily computed from a top-down camera, while the number of interactions is the episode length. To evaluate our policy, we load a task from the testing task datasets then run the policy for 10 steps or until the policy predicts grasps on the floor. The metrics are averaged across all tasks in the testing datasets.

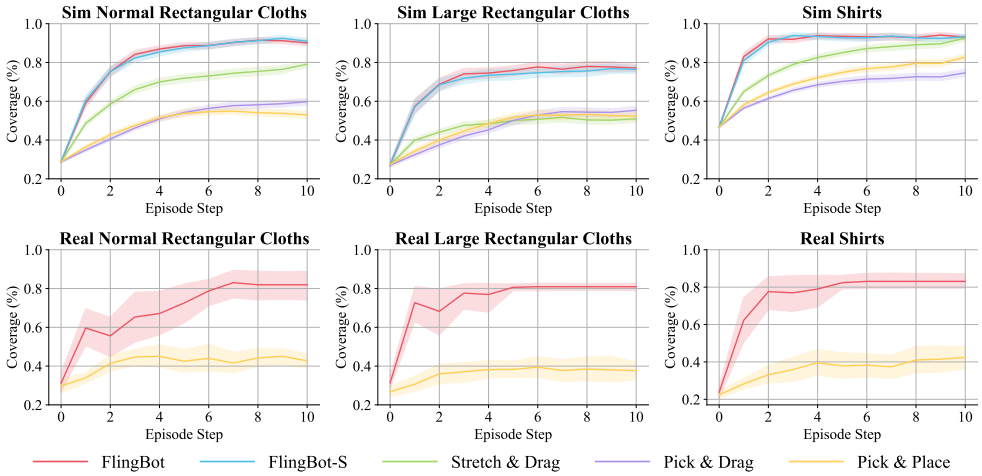

Figure 4: **Coverage v.s. Steps.** with 95% confidence interval shaded. [FlingBot] can achieve high coverage within few interactions step, while the quasi-static baselines never reach such a high coverage even with significantly more interaction steps. This demonstrates the difficulty of unfolding from highly crumpled initial configurations and dynamic action's superior efficiency in such challenging tasks.

4.2 Task Dataset Generation

Each task is specified by a cloth mesh, mass, stiffness, and initial configuration. The cloth mesh is sampled from one of three types:

1. **Normal Rect**, which contains rectangular cloths with size within the reach range. Edge lengths are sampled from [0.40m,0.65m]
2. **Large Rect**, which contains rectangular cloths with at least one edge larger than the reach range (0.70m). Otherwise, edge sizes are sampled from [0.40m,0.75m], which means the shorter edge could still be smaller than the reach range.
3. **Shirt**, which contains a subset of shirts sampled from CLOTH3D's [23] test split, all of which are resized to be within the reach range. These shirts include tank tops, crop tops, short and long sleeves.

The cloth mass is sampled from $[0.2\text{kg}, 2.0\text{kg}]$ and an internal stiffness from $[0.85\text{kg/s}^2, 0.95\text{kg/s}^2]$. Finally, the cloth's initial configuration is varied by holding a randomly grasped the cloth at a random height between [0.5m,1.5m] then dropping and allowing the cloth to settle (similar to [2] and tier-3 in [3]), resulting in a severely crumpled configuration.

To calculate the normalized coverage, we use the maximum possible coverage of the cloths in their flattened configurations. For rectangular cloths in simulation, the flattened configuration can be analytically calculated using the undeformed vertex positions of the grid mesh, which means the normalized coverage could still be higher if the cloth rests in a stretched position due to friction. For shirts in simulation, we opted to calculate its maximum possible coverage as its outer surface area divided by 2, since a qualitatively flattened T-shirt may not actually maximize coverage. While this choice also makes it possible for the normalized coverage to be greater than 1, it will still preserve performance rankings. For real world experiments, the flattened configurations are manually set.

We emphasize the difficulty of our unfolding tasks, where cloths in Normal Rect, Large Rect, and Shirt have average initial coverages of 28.8%, 27.1%, and 46.4% in simulation respectively, and 26.1%, 33.6%, and 24.0% in the real world respectively. In contrast, cloths in simulation from [3] start at 77.2%, 57.6%, and 42.0% for easy, medium, and hard tasks, while cloths in the real world from [16] starts at 71.4% and 68.4% on two different real-world trials. Yet, the challenging cloth configurations found in our tasks (Fig. 5) are realistic and prevalent in typical households.

In simulation, the policy is trained on 2000 *rectangular* cloths sampled evenly between Normal Rect and Large Rect, and evaluated on 600 novel tasks split evenly between Normal Rect, Large Rect, and Shirt cloths. In real, the simulation policy is deployed to collect real world experience on 150 Normal Rect cloth tasks, optimized on both simulation and real world data, then evaluated on 10 novel test tasks in each cloth type.

4.3 Approach Comparisons

- **Quasi-static**: [Pick & Place] and [Pick & Drag] are policies which uses a quasi-static "pick and place" (similar to Lee et al. [2], visualized in Fig. 2b) and "pick and drag" (similar to Seita et al. [24], with no lift step compared to pick and place) primitive respectively. [Stretch & Drag] is identical to [Pick & Drag] with

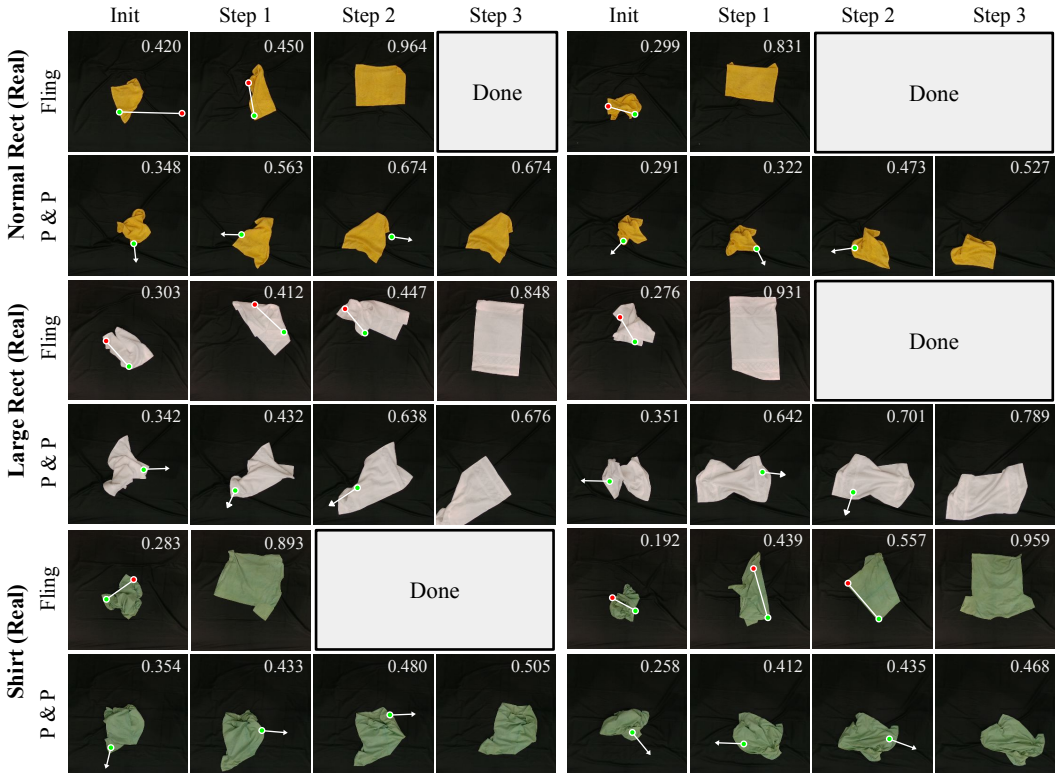

Figure 5: **Qualitative Results in Real World.** Cloth coverages are labeled on the top right corner. Red and green circles represent grasps by left and right arms, placed above and below field of view, respectively. FlingBot discovered through trial-and-error a two-arm corner and edge grasp when corners and edges are visible. While the pick & place baseline discovered a similar strategy, its performance is inherently limited by quasi-static actions, requiring significantly more steps to achieve a final coverage lower than FlingBot's.

an extra stretch step aftering picking, similar to FlingBot. These baselines are implemented with dual-arm set up, therefore provides the same physical reach range as our dual-arm fling system.

- **Dynamic manipulation**: [FlingBot] is our policy which predicts the optimal two-arm grasp locations for a fixed stretching and flinging routine described in Sec. 3.2.
- **Dynamic manipulation with fling speed prediction**: [FlingBot-S] is identical to [FlingBot] with an additional fling speed module, which predicts the fling speed within the range of $[0.1\text{ms}^{-1}, 1.0\text{ms}^{-1}]$ from visual input of the cloth after stretching and lifting. This fling speed module is trained using Deep Deterministic Policy Gradients (DDPG) [25] on the delta-coverage rewards to maximize single-step returns (discount factor γ is set to 0), where the grasps are predicted by a converged and frozen [FlingBot] value network. However, despite the cloth physical parameter variations in the training and testing dataset (Sec. 4.2), Tab. 1 and Fig. 4 suggest that there aren't significant performance gains for [FlingBot-S] over [FlingBot] within the cloth parameter ranges tested (Sec. 4.2). These results justify [FlingBot]'s usage of a single fling speed, even for varying cloth mass, stiffness, and size. Thus, we prefer the simpler [FlingBot] approach for comparisons with baselines.
- **Dynamic manipulation with fling parameter regression**: [Fling-Reg] is identical to [FlingBot], but directly predicts the action parameters $\langle C_x, C_y, \theta, w \rangle$ from visual inputs instead of exploiting the task's equivariances. Its policy is trained using DDPG on delta-coverage rewards to maximize single-step returns. However, from Tab. 1, [Fling-Reg] completely fails to perform the task, demonstrating the advantage of encoding inductive biases which leverage equivariances in the problem structure.

4.4 Results

Better Efficiency. In this experiment, we compare the unfolding efficiency between quasi-static and dynamic actions. From the Normal Rect column in Tab. 1, [FlingBot] increases the coverage of the cloth (+63.1%) by more than two times that of [Pick&Place] and [Pick&Drag] baselines (+29.2%, and +24.2%). Additionally, from Fig. 4, [FlingBot] achieves over 80% within 3 interactions (simulation normal cloth), while the quasi-static baselines never reach such a high coverage even with significantly more interaction steps or with a

stretching subroutine. This demonstrates the difficulty of unfolding from highly crumpled initial configurations and dynamic action's superior efficiency in such challenging tasks.

Increased Reach Range. In this experiment, we investigate these approaches' performance on cloths which have dimensions larger than the robot arm's reach range. These tasks are not only challenging for quasi-static baselines, because the arms can't manipulate the cloth at locations beyond its reach range, but also for our flinging policies, because the arms can't fully stretch or lift the cloth off the ground while flinging. Despite these challenges, [FlingBot] achieves 79.2% (Tab. 1, column Large Rect). Compared to the quasi-static baselines, [FlingBot] increases the coverage by +52.0%, which is roughly twice that of the quasi-static baselines (+27.1%, +24.8%, +23.1%). These results show how high-velocity actions could throw cloths to regions unreachable by quasi-static actions, allowing the system to be more flexible with extreme cloth sizes.

Generalize to Unseen Cloth Types. In this experiment, we investigate how well these approaches, trained on only rectangular cloths, can generalize to non-rectangular cloths such as different types of T-shirts. Qualitative real world (Fig. 5) and simulation (in supplementary materials) results suggest that our flinging policy has learned to grasp keypoints on cloth (i.e., corners, edges) when it sees them, or otherwise fling to reveal these features. FlingBot's generalization performance (93.3% in Tab. 1, column Shirt) to novel cloth geometries like shirts can be attributed to this strategy, since a cloth of any type can be unfolded by grabbing one of its edges, stretching, then flinging it in a direction perpendicular to the edge. Through self-supervised exploration, our approach discovered grasping strategies which were manually designed in heuristic based prior works, while being more generalizable to different cloth configurations and types. Meanwhile, all quasi-static baselines exhibited worse cloth unfolding efficiency. The Sim Shirts plot in Fig. 4 shows that our flinging policies take only 3 actions to reach their maximum coverages, while the quasi-static baselines take upwards of 8 steps to reach lower maximum coverages.

	Normal Rect	Large Rect	Shirt
Pick&Place	53.0 / 24.2	52.0 / 24.8	79.8 / 33.4
Pick&Drag	58.0 / 29.2	54.2 / 27.1	72.4 / 26.0
Stretch&Drag	77.2 / 48.4	50.2 / 23.1	90.3 / 43.9
Fling-Reg	29.1 / 0.3	27.7 / 0.5	54.0 / 7.6
FlingBot-S	**92.6 / 63.8**	78.9 / 51.7	**93.5 / 47.1**
FlingBot	91.9 / 63.1	**79.2 / 52.0**	93.3 / 46.8

Table 1: Simulation Experiments (Final / Delta Coverage).

Evaluating Real-World Unfolding. Finally, we finetune and evaluate our simulation models from Tab. 1 with real-world experience on a pair of UR5 arms. Task generation is automated using the robot arms by randomly grasp the cloth at height 1 m and drop it back on the workspace. We use a 0.35m $\times$ 0.45m cloth for Normal Rect, a 0.40m $\times$ 0.70m bath towel for Large Rect, and a 0.45m $\times$ 0.55m T-shirt for Shirt. The system collected 257 experience steps over 150 cloth tasks for finetuning in total. The performance is reported averaged over 10 test episodes, where real-world grasp errors are filtered out (see "Real World Failure Cases" below). From Tab. 2, we report that our policy achieves over 80% on all cloth types, which outperforms the quasi-static pick & place baseline by at least 40%. Additionally, we report that the pre-finetune performance of FlingBot on Normal Cloths is 69.8%, justifying our decision to finetune to get a 12.1% improvement. Overall, our flinging primitive takes a median time of 16.7s. While the pick & place primitive only takes a median time of 8.8s, it is unable to reach the same coverage our flinging primitive, even when taking many more interaction steps. Both primitives incur additional overheads of 2.5s for preparing the transformed observation batch and 2.9s for a routine which checks whether the cloth is stuck to the gripper after the gripper is opened (details in supplementary materials).

Action	Normal Rect	Large Rect	Shirt
Pick&Place	43.2 / 13.0	38.4 / 11.0	42.7 / 19.9
FlingBot	**81.9 / 55.8**	**88.5 / 54.9**	**89.2 / 65.2**

Table 2: Real World Experiment (Final / Delta Coverage).

Real World Failure Cases. Grasp failures - when the policy picks a grasp on the cloth, but the gripper fails to pinch grasp the cloth - make up most of the failure cases in FlingBot's real world system, and we discard episodes where grasp failures occur. Additionally, even on successful grasps, the gripper may hold the cloth tight in its crumpled state, rendering all flings ineffective. However, we do not discard of these episodes. We discuss more of real world grasp failures and show simulation failure cases in the supplementary material.

5 Conclusion and Future Work

We proposed a dynamic fling motion primitive and a self-supervised learning algorithm for learning the grasp parameters for the primitive of the cloth unfolding task. The cloth unfolding policy is efficient, generalizable, and works with cloth sizes beyond the reach range of the system in both simulation and the real world. While we've demonstrated the effectiveness of dynamic actions for cloth unfolding, dynamic actions alone are insufficient for more complex deformable object manipulation tasks, such as goal-conditioned folding. In this direction, future work could explore learning in combined action spaces (i.e., dynamic and quasi-static, two-arm and single-arm, etc.) for goal-conditioned cloth manipulation.

References

[1] M. T. Mason and K. M. Lynch. Dynamic manipulation. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, volume 1, pages 152–159 vol.1, 1993. doi:10.1109/IROS.1993.583093.

[2] R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner. Learning arbitrary-goal fabric folding with one hour of real robot experience, 2020.

[3] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, K. Yamane, S. Iba, J. F. Canny, and K. Goldberg. Deep imitation learning of sequential fabric smoothing policies. *CoRR*, abs/1910.04854, 2019. URL http://arxiv.org/abs/1910.04854.

[4] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel. Learning to manipulate deformable objects without demonstrations. *CoRR*, abs/1910.13439, 2019. URL http://arxiv.org/abs/1910.13439.

[5] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. 2018.

[6] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. 2019.

[7] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser. Spatial action maps for mobile manipulation. *Robotics: Science and Systems XVI*, Jul 2020. doi:10.15607/rss.2020.xvi.035. URL http://dx.doi.org/10.15607/RSS.2020.XVI.035.

[8] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

[9] X. Lin, Y. Wang, J. Olkin, and D. Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation, 2020.

[10] L. Sun, G. Aragon-Camarasa, W. Cockshott, S. Rogers, and J. Siebert. A heuristic-based approach for flattening wrinkled clothes. In *TAROS*, 2013.

[11] B. Willimon, S. Birchfield, and I. Walker. Model for unfolding laundry using interactive perception. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4871–4876, 2011. doi:10.1109/IROS.2011.6095066.

[12] D. Seita, N. Jamali, M. Laskey, R. Berenstein, A. K. Tanwani, P. Baskaran, S. Iba, J. F. Canny, and K. Goldberg. Robot bed-making: Deep transfer learning using depth sensing of deformable fabric. *CoRR*, abs/1809.09810, 2018. URL http://arxiv.org/abs/1809.09810.

[13] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. 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.

[14] D. Triantafyllou, I. Mariolis, A. Kargakos, S. Malassiotis, and N. Aspragathos. A geometric approach to robotic unfolding of garments. *Robotics and Autonomous Systems*, 75:233–243, 2016.

[15] H. Yuba, S. Arnold, and K. Yamazaki. Unfolding of a rectangular cloth from unarranged starting shapes by a dual-armed robot with a mechanism for managing recognition error and uncertainty. *Advanced Robotics*, 31(10):544–556, 2017.

[16] A. Ganapathi, P. Sundaresan, B. Thananjeyan, A. Balakrishna, D. Seita, J. Grannen, M. Hwang, R. Hoque, J. E. Gonzalez, N. Jamali, K. Yamane, S. Iba, and K. Goldberg. Learning dense visual correspondences in simulation to smooth and fold real fabrics, 2020.

[17] R. Jangir, G. Alenya, and C. Torras. Dynamic cloth manipulation with deep reinforcement learning. *arXiv preprint arXiv:1910.14475*, 2019.

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

[19] Y. Yamakawa, A. Namiki, and M. Ishikawa. Dynamic manipulation of a cloth by high-speed robot system using high-speed visual feedback. *IFAC Proceedings Volumes*, 44(1):8076–8081, 2011.

[20] M. Shibata, T. Ohta, and S. Hirai. Robotic unfolding of hemmed fabric using pinching slip motion. In *The Abstracts of the international conference on advanced mechatronics: toward evolutionary fusion of IT and mechatronics: ICAM 2010.5*, pages 392–397. The Japan Society of Mechanical Engineers, 2010.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation in robotics, games and machine learning, 2017.

[23] H. Bertiche, M. Madadi, and S. Escalera. Cloth3d: Clothed 3d humans. In *European Conference on Computer Vision*, pages 344–359. Springer, 2020.

[24] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng. Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks. In *ICAR*, 2021.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.