2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Dynamic Bimanual Cloth Manipulation via Dynamic Movement Primitives and Reinforcement Learning

Matija Mavsar, Aleš Ude, and Andrej Gams

***Abstract*— Learning effective control strategies for deformable object manipulation remains a major challenge in robotics, especially when tasks require fast, coordinated motions between multiple manipulators. In this work we address the problem of dynamic bimanual cloth manipulation, where two robotic arms must coordinate fast, fluid motions to place a cloth onto a surface. Our method uses Proximal Policy Optimization (PPO) with Dynamic Movement Primitives (DMPs) as the policy output, enabling smooth and parameterized trajectory generation. We compute rewards such as cloth height, corner alignment, and movement direction only at the end of each training episode, while the robot control continues at a higher frequency, and introduce a probing phase to obtain knowledge about cloth dynamics. We implement our approach in NVIDIA Isaac Sim with realistic cloth dynamics. Experiments show that this setup allows the robots to learn fast, coordinated bimanual cloth placement using only occasional reward feedback.**

I. INTRODUCTION

Robotic manipulation of deformable objects remains a challenging and underexplored problem, especially when tasks require high-speed, coordinated motions and contact-rich interactions. Unlike rigid-body manipulation, deformable object manipulation—such as folding, spreading, or positioning cloth—introduces nonlinear and unpredictable dynamics that are difficult to model or control directly [1], [2]. The complexity increases in bimanual settings, where synchronization between arms is critical, and the deformable object's configuration is constantly changing [3], [4].

In this work, we address dynamic bimanual cloth placement, where two robotic arms must learn to fling and place a cloth onto a surface through a fast, coordinated motion. This setting requires fine control over timing, directionality, and spatial alignment during the high-speed placement phase, making it a challenging benchmark for learning-based control strategies. Previous methods often focus on quasi-static tasks or rely on hand-crafted motion primitives [5], [6], whereas our approach aims to learn these motions dynamically using reinforcement learning.

We propose a reinforcement learning approach using Proximal Policy Optimization (PPO) [7], where the policy outputs parameters of Dynamic Movement Primitives (DMPs) [8], [9]. This formulation combines the strengths of reinforcement learning with the smoothness and adaptability of DMPs, enabling the robot to generate structured and parameterized trajectories suited for deformable manipulation. Our DMP-based policy operates in a continuous action space and supports real-time trajectory generation.

All authors are with Jožef Stefan Institute, Dept. of Automatics, Biocybernetics, and Robotics, Ljubljana, Slovenia. matija.mavsar@ijs.si, andrej.gams@ijs.si, ales.ude@ijs.si

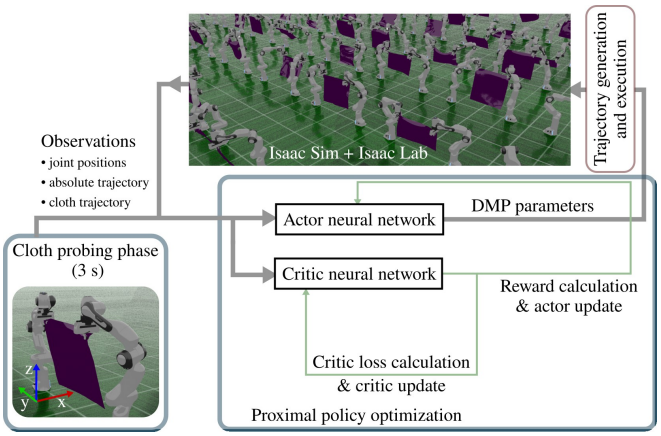

Fig. 1. The proposed method. Observations from a probing phase, along with the rewards from the resulting motion, are used to update actor and critic network using proximal policy optimization. Actor's output is employed to generate and execute cloth placement trajectories in *Isaac Sim* with the *Isaac Lab* framework. 64 replicas are run in parallel on a single GPU during training.

To improve training efficiency and task performance, we design a composite reward function incorporating multiple criteria, including height reward to encourage full contact between the cloth and the surface, corner reward to incentivize flinging of the cloth away from the robots, and direction reward to guide the cloth's orientation when placed.

Additionally, inspired by human behavior, we introduce a probing phase to observe the cloth dynamics before executing the placement motion, improving the agent's adaptability to different cloth types.

Our contributions include:

- A PPO-based learning framework for dynamic bimanual cloth placement with structured policy outputs via DMPs, enhanced by a probing phase to "test" the cloth.
- A multi-component reward function tailored to dynamic cloth manipulation, with sparse rewards, meaning we only compute rewards at the end of episodes.
- An implementation in Isaac Sim and experimental results demonstrating placement accuracy and generalization across different cloth dimensions.

II. RELATED WORK

Robotic manipulation of deformable objects has accelerated in recent years, yet much of the literature still targets slow, single-arm actions or relies on handcrafted motion primitives. Below, we discuss the most relevant research and explain how our work fits into this context.

978-1-6654-5784-2/26/$31.00 ©2026 IEEE

A. Benchmarks and Simulators for Deformable Objects

Standardised suites such as SOFTGYM established the first large-scale benchmark for cloth, rope and fluid manipulation, and revealed the sample inefficiency of deep RL on deformables [1]. Subsequent efforts ported particle-based cloth to GPU-accelerated engines such as Isaac Sim, but existing tasks remain largely quasi-static, i.e., without engaging any transient dynamics. Our work extends this simulation line by tackling *dynamic* bimanual cloth placement rather than the slower draping or folding scenarios that dominate current benchmarks.

B. Bimanual Cloth Manipulation

Early dual-arm systems achieved cloth folding or unfolding with scripted motion planning and visually guided state machines [3]. More recent learning approaches include FABRICFLOWNET, which couples optical-flow perception with a policy that switches between single- and dual-arm pick-and-place actions [4], and GRAPHGARMENT, which learns a graph-based dynamics model for planning bimanual trajectories [10]. These methods emphasise geometric precision but operate in low-speed pick–place regimes. In contrast, our method learns *continuous, high-speed* dual-arm motions directly through reinforcement learning.

C. Dynamic and High-Speed Strategies

Momentum-rich manoeuvres have proved surprisingly effective for cloth manipulation. FLINGBOT learns to unfold crumpled fabrics by stretching and flinging [5], while SPEEDFOLDING attains state-of-the-art throughput in garment folding via a learned library of bimanual primitives [6]. Both rely on engineered primitive libraries or supervised pick-point predictors and focus more on optimal pickup locations, whereas we focus on discovering optimal dynamic flinging strategies using end-to-end RL.

D. Reinforcement Learning for Deformable Objects

Sparse-reward RL has been explored for dynamic napkin placement in the SOFA simulator [11], however, it relies on demonstrations and does not investigate the placement of different napkin sizes. An earlier study by Tsurumine *et al.* introduced deep reinforcement learning with smooth policy update, achieving real-robot cloth flipping and T-shirt folding with fewer than 1000 samples, but the demonstrations remained quasi-static and single-arm [2]. Variable-stiffness humanoids have also been explored for cloth swinging [12], employing a brief randomized probing phase to infer cloth dynamics—closely aligned with our premise. However, their approach relies on continuous, vision-in-the-loop computation of joint targets from images.

E. Dynamic Movement Primitives in RL

Dynamic Movement Primitives (DMPs) provide smooth, time-scalable motion representations for various dynamic tasks [9] and have recently been embedded in policy networks to boost sample efficiency. Li *et al.* combine DMPs with actor–critic RL for general robot manipulation [13], while CLOTHPPO applies a similar idea to quasi-static cloth unfolding [14]. Recent works highlight DMPs' advantages for trajectory generalization and control stability [15], [16]. Our policy builds on this paradigm by generating complete Cartesian-velocity DMPs at the start of each episode. This approach simplifies motion representation and allows for fast generalization through parameter changes, e.g., goal position.

F. Contribution Beyond Existing Work

Previous work has either (i) executed dynamic cloth actions with heuristics, (ii) learned bimanual cloth skills in quasi-static settings, or (iii) combined RL with DMPs for rigid-body tasks. We unify these approaches by **(a)** learning dynamic dual-arm cloth placement entirely from RL, exploring the effect of probing the cloth before learning, **(b)** embedding DMPs in a PPO actor for real-time smooth trajectory generation, and **(c)** introducing a sparse-evaluation reward schedule that cuts simulation cost.

III. BACKGROUND

Proximal Policy Optimization (PPO) [7] is a popular reinforcement learning algorithm that improves training stability by using a clipped objective function. At each time step t, the agent observes a state $\mathbf{s}_t$, selects an action $\mathbf{a}_t$, and receives a reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$. PPO limits the change in the policy during each update, balancing exploration and exploitation. The objective is:

$$L^{\text{clip}}(\boldsymbol{\theta}) = \mathbb{E}_t \left[\min \left(\rho_t(\boldsymbol{\theta}) \hat{A}_t, \text{clip}(\rho_t(\boldsymbol{\theta}), 1-\varepsilon, 1+\varepsilon) \hat{A}_t \right) \right], \tag{1}$$

where $\rho_t(\boldsymbol{\theta})$ is the probability ratio between the current and previous policy, $\hat{A}_t$ is the advantage estimate. The clip function makes sure that $\rho_t(\boldsymbol{\theta})$ is between $1-\varepsilon$ and $1+\varepsilon$. The advantage $\hat{A}_t$ is typically computed using the Generalized Advantage Estimate (GAE), given by:

$$\hat{A}_t = \sum_{l=0}^{T-1} (\gamma\lambda)^l \delta_{t+l}, \tag{2}$$

where $\delta_t = r_t + \gamma V(\mathbf{s}_{t+1}) - V(\mathbf{s}_t)$, $V(\mathbf{s})$ is the value function, $\gamma \in [0,1)$ is the discount factor, and λ balances stability and accuracy; lower values yield smoother but biased estimates, while higher values reduce bias at the cost of higher variance.

In our work we employ PPO to achieve dynamic flinging motions using two robots. We adapt this algorithm for our task and compute the advantage function in a modified way. Further details can be found in Section IV-F.

IV. ROBOT CONTROL FOR BIMANUAL CLOTH MANIPULATION

In this section we detail how *Dynamic Movement Primitives* (DMPs) are employed to generate and adapt dual-arm trajectories during reinforcement learning of cloth manipulation skills. At the start of every control cycle the policy (represented by an actor network $\pi_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$) outputs a fresh set of DMP parameters. These parameters are integrated at each simulator physics step, yielding Cartesian set-points for both manipulators; rewards are then computed

from the resulting cloth state (Section IV-G). By embedding DMPs in this loop we obtain smooth, time-scalable motions while keeping the action dimension compact.

A. Representation Using Absolute and Relative Coordinates

For bimanual cloth manipulation we adopt an absolute/relative coordinate scheme [17], [18]. The absolute position $\mathbf{p}_a$ of the robot system can be defined as the midpoint between end-effector positions $\mathbf{p}_1$ and $\mathbf{p}_2$, while the absolute orientation $\mathbf{R}_a$ is defined as the orientation that is halfway between the orientations of both end effectors:

$$\mathbf{p}_a = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2}, \quad \mathbf{R}_a = \mathbf{R}_1 \mathbf{R}^1_{\mathbf{k}^1_{12}}(\kappa_{12}/2). \tag{3}$$

More specifically, $\mathbf{k}^1_{12}$ and κ_{12} are the unit vector and the angle that realize the rotation from the orientation of the first end effector to the orientation of the second one. Therefore, $\mathbf{R}^1_{\mathbf{k}^1_{12}}(\kappa_{12}/2)$ corresponds to a rotation about axis $\mathbf{k}^1_{12}$ by half of an angle needed to align both end effectors.

Relative position $\mathbf{p}_r$ and relative orientation $\mathbf{R}_r$ are defined as the positional and rotational difference between one of the end effectors relative to another, $\mathbf{p}_r = \mathbf{p}_2 - \mathbf{p}_1$, $\mathbf{R}_r = \mathbf{R}^2_1$.

We control the robot by sending the desired absolute and relative positions and perform inverse kinematics with the procedure described in [17]. We construct a Jacobian matrix $\mathbf{J}$ from an absolute Jacobian matrix $\mathbf{J}_a$ and a relative Jacobian matrix $\mathbf{J}_r$, where

$$\mathbf{J}_a = [\tfrac{1}{2}\mathbf{J}_1 \quad \tfrac{1}{2}\mathbf{J}_2], \quad \mathbf{J}_r = [-\mathbf{J}_1 \cdot \mathbf{J}_2], \quad \mathbf{J} = \begin{bmatrix}\mathbf{J}_a \\ \mathbf{J}_r\end{bmatrix}, \tag{4}$$

with $\mathbf{J}_1$ and $\mathbf{J}_2$ being Jacobian matrices for the two robot arms. The resulting matrix $\mathbf{J} \in \mathbb{R}^{12\times 14}$ is then used together with the error $\mathbf{e} = [\mathbf{e}_a^T, \mathbf{e}_r^T]^T$, comprising absolute and relative errors $\mathbf{e}_a$ and $\mathbf{e}_r$, respectively, to calculate inverse kinematics.

This sort of dual-arm trajectory representation naturally couples the two arms and reduces the number of parameters the policy must output.

B. Dynamic Movement Primitives

We encode Cartesian robot trajectories using DMP parameters [19], which are output by the policy. In a Cartesian space DMP, the robot's motion is specified by its position $\mathbf{p}(t) \in \mathbb{R}^3$ trajectory. In the second-order DMP system, the trajectory can be described by the following system of differential equations:

$$\tau \mathbf{z} = \alpha_z(\beta_z(\mathbf{g} - \mathbf{p}) - \mathbf{z}) + \mathrm{diag}(\mathbf{g} - \mathbf{p}_0)\,\mathbf{f}(x), \tag{5}$$

$$\tau \dot{\mathbf{y}} = \mathbf{z}. \tag{6}$$

Here $\mathbf{p}_0 \in \mathbb{R}^3$ represents the initial position of the desired trajectory and $\mathbf{g} \in \mathbb{R}^3$ is the final position or goal of the trajectory and $\mathrm{diag}(\mathbf{g} - \mathbf{p}_0) \in \mathbb{R}^{3\times 3}$ is a matrix with components of the vector $\mathbf{g} - \mathbf{p}_0$ on the diagonal. The auxiliary parameter $\mathbf{z} \in \mathbb{R}^3$ is the scaled velocity of motion, obtained with the temporal scaling term τ, and $\mathbf{f}(x) \in \mathbb{R}^3$ denotes a nonlinear forcing term, where $x \in \mathbb{R}$ is the phase defined by

$$\tau \dot{x} = -\alpha_x x. \tag{7}$$

The forcing term $\mathbf{f}(x)$ is a linear combination of normalized radial basis functions (RBFs), defined with weights, and can thus be used to approximate an arbitrary function. In this way, the dynamic system can reproduce any smooth motion between the initial and final robot position. Note that if switching of DMP parameters during the motion is desired, third-order DMPs should be used, as shown in [20].

C. DMP-Based Trajectory Generation

Each control cycle begins with the actor emitting DMP parameters that define second-order Cartesian DMPs for the absolute and relative channels.

While the general second–order formulation in Eqs. (5)–(7) assumes a full 3-D Cartesian position $\mathbf{y}(t) \in \mathbb{R}^3$, our cloth–placement task requires control only in the horizontal (x) and vertical (z) directions of the *absolute* frame; the y coordinate of the absolute trajctory is set to zero. We therefore only predict a 2-D absolute trajectory in the x-z plane. The relative trajectory is fixed so that the grippers of the two robots are always 0.7 m apart in the (x)-axis and are facing downward with a constant 180 degree rotation around z-axis from the first gripper to the second (note that the robots are facing each other as shown in Fig. 1).

With $M = 25$ radial-basis kernels (defined empirically), the parameter vector produced by the actor network $\pi_{\boldsymbol{\theta}}$ is

$$\mathbf{a} = \left[\mathbf{y}_0,\, \mathbf{g},\, \{\mathbf{w}_i\}_{i=1}^{25},\, \tau\right] \in \mathbb{R}^{55}, \tag{8}$$

where $\mathbf{y}_0, \mathbf{g} \in \mathbb{R}^2$ (4 scalars), $\mathbf{w}_i \in \mathbb{R}^2$ (50 scalars), i.e. the weights of the forcing term in Eq. (5), and the duration $\tau \in \mathbb{R}$ (1 scalar). During roll-out the planar DMP is integrated at 120 Hz to yield the absolute x–z trajectory, with y being set to 0 (the midpoint between two end effectors). Using the procedure in Section IV-A we obtain control joint positions for both robots.

By representing dual-arm actions as DMP parameters in absolute/relative space, we generate smooth, low-dimensional commands for high-quality cloth manipulation. The policy updates weights at the start of each cycle, with the continuous integration of these weights guiding the motion and generating reward signals from the cloth's behavior.

D. One-shot Learning After Probing

Motivated by the observation that most humans briefly shake or dip a fabric to gauge weight, stiffness and damping before committing to a full placement motion, we begin training by executing a 3-second probing phase. During this phase the bimanual agent performs a predefined motion that excites the cloth's nodes, allowing the critic to observe corner displacements that implicitly encode material parameters. Training is thus split into parts:

1) **Probing phase (0–3 s).** Both arms follow a simple predefined trajectory at a physics step of $1/120$ s. The motion excites the cloth so the agent can observe corner oscillations.

2) **Placement phase (3 s -).** After the probing phase, we train in 3-second episodes. In each episode, observations from the probing phase (absolute position trajectory, corner trajectory) are downsampled and flattened into a one-dimensional vector and fed into the policy together with current joint positions and velocities.

The policy produces a 55-dimensional DMP parameter vector $\mathbf{a}$ from Eq. (8) that defines a 3 s placement trajectory; the policy is *not* queried again during that interval. We integrate the DMP and obtain absolute and relative positions (see Sections IV-A and IV-B).

This "single-shot" decision mirrors a human who probes the cloth, decides on a throw, and then commits without mid-flight corrections.

E. Observations and Rewards

We utilize the GPU–centric implementation of Proximal Policy Optimization (PPO) from the RSL–RL [21] library, part of the Isaac Lab API [22]. Both actor and critic are 3-layer multilayer perceptrons with hidden sizes $[256, 128, 64]$ and ELU (exponential linear unit) activations.

At the beginning of each episode the actor $\pi_{\boldsymbol{\theta}}$ and critic $V_{\boldsymbol{\phi}}$ receive a 92-dimensional observation (or state) vector $\mathbf{s}_t \in \mathbb{R}^{92}$ comprising

- current joint position and velocities of both Panda arms $(2\times 7+2\times 7=28)$;
- Cartesian poses of the absolute frame, gathered during the 3-second probing phase, sampled at 2.5 Hz (x, z only, 2 axes × 8 time steps $= 16$);
- positions of the two free corners positions, also gathered during the 3-second probing phase, sampled at 2.5 Hz (2 corners × 3 axes × 8 time steps $= 48$).

F. Training Formulation

In our approach, we modify the standard PPO algorithm by using a simplified variant where observations are collected only once at the beginning of the training process. The agent first performs a probing phase, where it collects initial observations and gathers implicit information about the cloth's dynamic properties. Following this, the training episodes are carried out. At the beginning of each training episode, the observations are input into the policy network, which generates Dynamic Movement Primitive (DMP) parameters that define the robot's trajectory.

Unlike traditional PPO implementations, where policy updates occur after each timestep based on intermediate rewards, our approach decouples control execution from policy optimization. In this framework, the agent executes the entire trajectory based on the generated DMP parameters, and only the final reward, calculated at the end of the motion, is used for the policy update. This approach is well-suited to our scenario since we are primarily concerned with the final outcome: the state of the cloth after the placement motion, rather than intermediate rewards. By removing intermediate updates and focusing solely on the final reward, this modification enables more stable and efficient learning.

a) Advantage estimate: In our formulation, we compute the generalized advantage estimate only once, at the end of the episode, using the final reward:

$$\hat{A}_t = r_T - V_{\boldsymbol{\phi}}(\mathbf{s}_0), \tag{9}$$

where r_T is the total reward calculated at the end of the episode as detailed in Section IV-G, $V_{\boldsymbol{\phi}}(\mathbf{s}_0)$ is the state-value function predicted by the critic network for the starting state $\mathbf{s}_0$, and $\boldsymbol{\phi}$ represent critic parameters. Since the reward is only calculated at the end, the advantage is the difference between the final reward and the initial value prediction for that episode.

b) Clipped-surrogate objective: The actor parameters $\boldsymbol{\theta}$ are updated by maximizing the following objective:

$$\begin{aligned} L^{\pi}(\boldsymbol{\theta}) \;=\; & \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)}\Big[\min\big(\rho_t(\boldsymbol{\theta})\hat{A}_t, \\ & \operatorname{clip}(\rho_t(\boldsymbol{\theta}), 1\pm\varepsilon)\hat{A}_t\big)\Big], \end{aligned} \tag{10}$$

where $\rho_t(\boldsymbol{\theta}) = \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)/\pi_{\boldsymbol{\theta}_{\text{old}}}(\mathbf{a}_t|\mathbf{s}_t)$ is the likelihood ratio between the new and old policy, and $\varepsilon = 0.2$.

c) Critic loss and total objective: The critic loss is computed to minimize the difference between the predicted value $V_{\boldsymbol{\phi}}(\mathbf{s}_t)$ and the final reward r_T using:

$$\begin{aligned} L^{\mathrm{V}}(\boldsymbol{\phi}) \;=\; & \mathbb{E}_{\mathbf{s}_t}\Big[\max\big((V_{\boldsymbol{\phi}}(\mathbf{s}_t) - r_T)^2, \\ & \big(\operatorname{clip}(V_{\boldsymbol{\phi}}(\mathbf{s}_t), V_{\boldsymbol{\phi}_{\text{old}}}(\mathbf{s}_t)\pm\varepsilon) - r_T\big)^2\big)\Big]. \end{aligned} \tag{11}$$

The full loss per minibatch is defined as $L = -L^{\pi} + c_v L^{\mathrm{V}} - c_e S[\pi_{\boldsymbol{\theta}}]$, where a higher $c_v = 1.0$ results in lower-variance advantages and often more stable updates, while $c_e = 0$ increases policy entropy $S[\pi_{\boldsymbol{\theta}}]$, increasing exploration. We set $c_v = 1.0$ and $c_e = 0$ and the loss function simplifies to $L = -L^{\pi} + L^{\mathrm{V}}$.

G. Reward Evaluation

We define four task-specific rewards that guide the policy towards high-quality cloth placement:

- **Spread reward** r_{area} — encourages a large Euclidean distance between each corner and every other corner $\mathbf{a}_i$, $\mathbf{a} \in \mathbb{R}$, $i = 0, \ldots, 3$, reducing wrinkles:

$$r_{\text{area}} = \sum_{i=0}^{3}\sum_{j=i+1}^{3} \|\mathbf{a}_i - \mathbf{a}_j\|_2. \tag{12}$$

- **Height reward** r_{height} — penalizes the mean vertex height of the cloth, driving the fabric down onto the surface:

$$r_{\text{height}} = \frac{1}{1+\overline{h}}, \tag{13}$$

  with $\overline{h}$ being the mean height (z value) of cloth vertices.
- **Corner–X reward** $r_{\text{corner-x}}$ — rewards large positive x–distance between each free (ungrasped) corner and the robot base frame, promoting an outward fling:

$$r_{\text{corner-x}} = \left(\frac{1}{2}\sum_{i=2}^{3}\left(\mathbf{a}_{x,i} - x_{\text{origin}}\right)\right) \tag{14}$$

- **Direction reward** $r_{\text{direction}}$ — Encourages a large difference in x-distance between free and grasped corners, promoting the spread of the cloth, as well as an outward fling:

$$r_{\text{direction}} = \left(\frac{1}{2} \sum_{i=2}^{3} \left(\mathbf{a}_{x,i} - x_{\text{origin}} \right) - \frac{1}{2} \sum_{i=0}^{1} \left(\mathbf{a}_{x,i} - x_{\text{origin}} \right) \right) \tag{15}$$

The composite reward is given by:

$$r_T = \gamma_a \, r_{\text{area}} + \gamma_h \, r_{\text{height}} + \gamma_x \, r_{\text{corner-x}} + \gamma_d \, r_{\text{direction}}, \tag{16}$$

where $\gamma_a, \gamma_h, \gamma_x, \gamma_d$ are weighting factors, which determine how much emphasis should be put on each reward. They were set to $1, 1, 10, 15$, respectively, and were defined empirically through testing. Changing these parameters can significantly influence the time it takes for the robot to learn the flinging motion, since it may focus too much on one reward. Rewards are evaluated after the agent executes the full DMP motion trajectory.

Furthermore, at each control timestep, we monitor the distance between the end effectors of both robots, ensuring it stays within an optimal range. We also track the height of the end effectors to prevent them from reaching dangerously low levels, which could result in a collision with the ground. If either condition is violated, we reduce the total reward r_T by 5 at the end of the episode, discouraging such behavior.

V. Experiments

We adopt the Proximal Policy Optimization algorithm and the highly optimized RSL–RL implementation [21], which is part of *NVIDIA Isaac Lab* [22]. Training runs inside *Isaac Lab*, which provides a thin wrapper over *Isaac Sim* for rapid environment creation and domain randomization. The cloth is defined as *Isaac Sim*'s particle cloth object, with bend stiffness set to 500, spring damping to 0.2, shear stiffness to 200, and stretch stiffness to 10^6. Simulation of robot environments, policy inference and policy training are executed on a single RTX 4090 GPU. Policies are trained for $16,000$ episodes, where the first episode is the initial probing phase, while the rest are placement episodes, as explained in Section IV-D.

The cloth's width is fixed at 0.7 m, while its length is sampled uniformly from $[0.45, 0.7]$ m in every environment, forcing the policy to generalize across sizes. Two Franka Emika Panda arms grasp neighboring corners. Each environment steps at 120 Hz and terminates when 3 s elapse.

We compare our training regime against two baselines:

- **Static-start**: Identical setup, but no probing is performed. All observations are set to zero and are therefore constant time-wise.
- **Single-length**: All environments use the same 0.7×0.7 m square cloth. During training, the policy can therefore not generalize to different lengths, meaning that it may perform worse at test time.

These baselines isolate the effect of the exploratory phase and of length randomization on final placement performance.

A. Training Hyper-Parameters

From the 64 parallel environments, we gather 16 steps each (2048 transitions) before every update. The clipped-surrogate objective uses $\varepsilon = 0.2$ to prevent large policy changes and the same value is used to curb large critic updates. Separately, an adaptive Kullback-Leibler (KL) schedule lowers the learning rate (initially 5×10^{-4}) whenever the empirical KL divergence exceeds the target 0.01.

Each update performs 8 epochs over 8 mini-batches, clips gradient norms at 1.0, and assigns equal weight to policy and value losses ($c_v = 1$). A complete run of $16,000$ PPO iterations finishes in roughly 60 hours on an NVIDIA RTX 4090.

VI. Results & Discussion

We evaluates the learned policy on placing of cloths with lengths between 0.45 and 0.70 m and compare it with two baselines, **Static-start** and **Single-length**. Evaluation metrics used were:

- **Reward return**: rewards defined in Section IV-G.
- **Coverage** (%): ratio between the cloth area on the ground and the cloth area when completely unfolded.

A video demonstrating the flinging motion is available as supplementary material for the paper.

A. Learning Curves

Fig. 2 presents the smoothed episode return over $16,000$ PPO iterations. As expected, the highest rewards during training are observed with the single-length cloth training, as this setup is simpler for the policy to learn. Both the static-start and full methods achieve similar reward levels, with the exception of the "corner-X" reward, where the full method significantly outperforms the static-start method. This means that the information from the probing phase enables the policy to fling the cloth further away regardless of the cloth length. However, it is important to note that these training rewards do not necessarily correlate with performance on previously unseen cloths of different lengths, as demonstrated in Section VI-B.

B. Quantitative Comparison on Unseen Cloths

Table I presents the rewards for the three trained methods evaluated on 500 previously unseen cloths. The results indicate that the highest rewards are achieved using the full method, which involves training with varying cloth lengths and an initial probing motion. The static-start method performs slightly worse but still relatively well, suggesting that training without the probing phase can still lead to effective cloth manipulation across different cloth lengths based solely on rewards.

The worst-performing method was single-length, as expected, since it was trained on identical cloths and lacked the ability to generalize to different cloth sizes.

These findings are further supported by the coverage results in Table II, where the full method achieves the best coverage, indicating the greatest success in spreading the cloth flat on the ground. The static-start method also

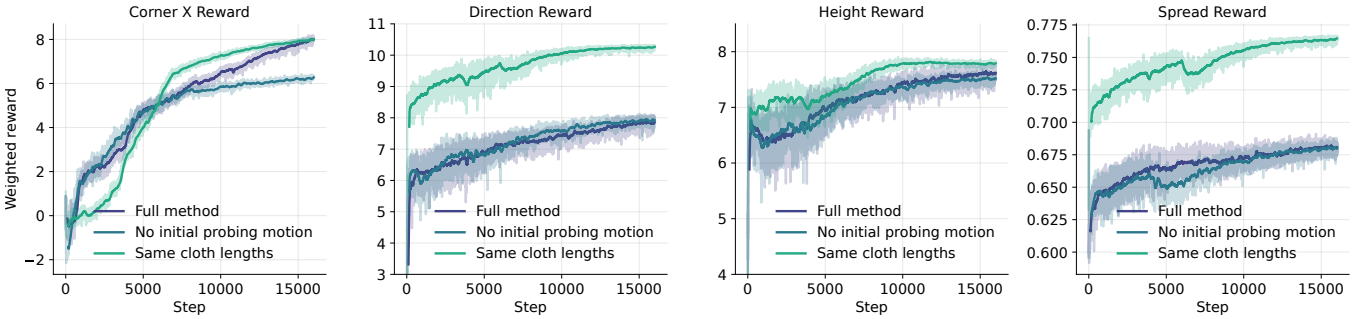

Fig. 2. Running average episode rewards during training (solid lines) for different methods, shaded lines showing actual values of rewards. All methods were trained for 16,000 episodes using the same hyperparameters.

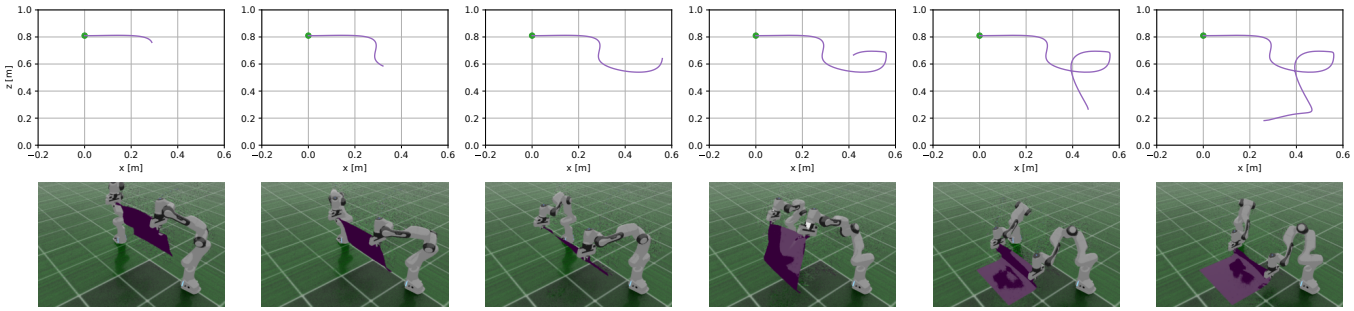

Fig. 3. Example cloth placing trajectory and corresponding camera frames during cloth placement. First row: absolute end-effector trajectory in x-z plane, generated by the planar DMP (one rollout). Second row: corresponding camera frames at $t = \{0.5, 0.75, 1.0, 1.33, 1.83, 2.5\}$ s.

TABLE I

REWARDS WITH **UNSEEN** CLOTH LENGTHS (0.45–0.7 M). MEAN ± STANDARD DEVIATION OVER 500 ENVIRONMENTS.

Method	r_{area}	r_{height}	$r_{\text{corner-x}}$	$r_{\text{direction}}$
Static-start	0.68 ± 0.04	7.49 ± 0.11	6.50 ± 0.71	7.95 ± 1.12
Single-length	0.58 ± 0.1	6.34 ± 1.39	6.82 ± 1.45	3.75 ± 3.53
Full	$\mathbf{0.69 \pm 0.04}$	$\mathbf{7.69 \pm 0.14}$	$\mathbf{8.2 \pm 0.59}$	$\mathbf{8.05 \pm 1.13}$

Note: The values represent the rewards for each method over 500 environments. Cloth lengths were randomly selected but were the same for each method.

TABLE II

COVERAGE ON **UNSEEN** CLOTH LENGTHS (0.45–0.7 M). MEAN ± STANDARD DEVIATION OVER 500 ENVIRONMENTS.

Method	Coverage [%]	% of envs. with coverage $> 80\%$
Static-start	92.89 ± 3.17	99.41
Single-length	65.5 ± 19.7	27.54
Full	$\mathbf{94.77 \pm 2.18}$	**99.61**

Note: Coverage denotes the ratio between the placed cloth and the ideally placed cloth, based on its dimensions. The second column gives the percentage of environments where coverage exceeds 80%.

demonstrates high coverage, though slightly lower than the full method.

C. Trajectory Visualisation

First row in Fig. 3 plots the absolute trajectory of the robots in the x-z plane. Corresponding snapshots in the second row illustrate how the cloth inverts mid-air and lands flat with minimal wrinkles.

Additional trajectories for four distinct environments are shown in Fig. 4, along with top-down views of the final cloth configurations after the flinging motion. As observed, all cloths land flat on the surface and are positioned far from the robots, resulting in a high "corner-X" reward. This indicates that the policy effectively adapts the motion based on the dynamic observations gathered during the probing phase, even when the cloth lengths vary.

VII. CONCLUSION

We presented an end-to-end reinforcement learning approach for dynamic bimanual cloth manipulation and came to the following conclusions:

- The probing phase at the beginning of training allows the agent to infer cloth dynamics and adapt to various fabric sizes.
- Combining sparse reward evaluation with varying cloth lengths during training further enhances the policy's ability to generalize to unseen cloth configurations.
- Training on a single cloth size or without the probing phase results in lower performance, showing importance of both generalization and dynamic cloth interaction.

Based on this we conclude that our method has potential for use in several high-speed dynamic cloth manipulation tasks. Future work will focus on enabling sim-to-real transfer and extending the framework to handle more complex tasks such as folding or contact-rich manipulation. Additionally, refining the probing phase could further enhance generalization to different cloth types and sizes.

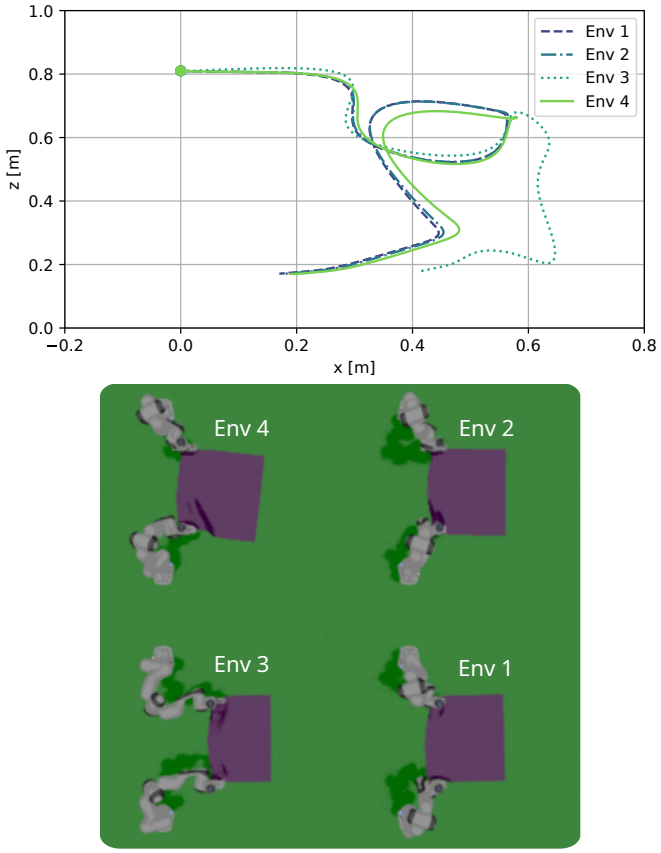

Fig. 4. Cloth placement absolute trajectories in x-z plane for 4 different environments (top) along with the final image of all environments at the end of placing motion (bottom). All cloths are almost completely spread. Environment nr. 3 has a distinct trajectory due to the much shorter length of the cloth.

ACKNOWLEDGMENT

This work has received funding from the program group Automation, robotics, and biocybernetics (P2-0076), supported by the Slovenian Research Agency, from DIGI-TOP, GA no. TN-06-0106, funded by Ministry of Higher Education, Science and Innovation of Slovenia, Slovenian Research and Innovation Agency, and European Union – NextGenerationEU, and from EU's Horizon Europe grants euRobin (GA no. 101070596), ROMANDIC (GA no. 101159522) and FlexCycle (GA no. 101189600).

REFERENCES

[1] X. Lin, Y. Wang, J. Olkin, and D. Held, "SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 155, 2021, pp. 432–448.

[2] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robotics and Autonomous Systems*, vol. 112, pp. 72–83, 2019.

[3] K. Yamazaki, R. Oya, K. Nagahama, K. Okada, and M. Inaba, "Bottom dressing by a dual-arm robot using a clothing state estimation based on dynamic shape changes," *International Journal of Advanced Robotic Systems*, vol. 13, no. 1, p. 5, 2016.

[4] T. Weng, S. M. Bajracharya, Y. Wang, K. Agrawal, and D. Held, "Fabricflownet: Bimanual cloth manipulation with a flow-based policy," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 192–202.

[5] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 164, 2022, pp. 24–33.

[6] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, and K. Goldberg, "Speedfolding: Learning efficient bimanual folding of garments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1–8.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[8] S. Schaal, J. Peters, J. Nakanishi, and A. J. Ijspeert, "Learning movement primitives," in *Robotics Research, The Eleventh International Symposium*, 2005, pp. 561–572.

[9] B. Nemec and A. Ude, "Action sequencing using dynamic movement primitives," *Robotica*, vol. 30, no. 5, pp. 837–846, 2012.

[10] W. Chen, K. Li, D. Lee, X. Chen, R. Zong, and P. Kormushev, "GraphGarment: Learning garment dynamics for bimanual cloth manipulation tasks," *CoRR*, vol. abs/2503.05817, 2025.

[11] R. Jangir, G. Alenyà, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4630–4636.

[12] K. Kawaharazuka, A. Miki, M. Bando, K. Okada, and M. Inaba, "Dynamic cloth manipulation considering variable stiffness and material change using deep predictive model with parametric bias," *Frontiers in Neurorobotics*, vol. 16, p. 890695, 2022.

[13] S. Li, W. Huang, C. Miao, K. Xu, Y. Chen, T. Sun, and Y. Cui, "Efficient robot manipulation via reinforcement learning with dynamic movement primitives-based policy," *Applied Sciences*, vol. 14, no. 22, p. 10665, 2024.

[14] L. Yang, Y. Li, and L. Chen, "ClothPPO: A proximal policy optimization enhancing framework for robotic cloth manipulation with observation-aligned action spaces," *CoRR*, vol. abs/2405.04549, 2024.

[15] R. Pahič, Z. Lončarević, A. Gams, and A. Ude, "Robot skill learning in latent space of a deep autoencoder neural network," *Robotics and Autonomous Systems*, vol. 135, p. 103690, 2021.

[16] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[17] P. Chiacchio, S. Chiaverini, and B. Siciliano, "Direct and inverse kinematics for coordinated motion tasks of a two-manipulator system," *Journal of Dynamic Systems, Measurement, and Control*, vol. 118, no. 4, pp. 691–697, 1996.

[18] B. Nemec, N. Likar, A. Gams, and A. Ude, "Bimanual human robot cooperation with adaptive stiffness control," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Cancun, Mexico, 2016, pp. 607–613.

[19] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.

[20] M. Mavsar, B. Ridge, R. Pahič, J. Morimoto, and A. Ude, "Simulation-aided handover prediction from video using recurrent image-to-motion networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 494–506, 2024.

[21] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proceedings of the 5th Conference on Robot Learning*, vol. 164. PMLR, 2022, pp. 91–100.

[22] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.