IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.
Cite as T-ASE paper.

# Robot Deformable Object Manipulation via NMPC-Generated Demonstrations in Deep Reinforcement Learning

Haoyuan Wang, *Graduate Student Member, IEEE*, Zihao Dong, Tong Zhu, Hongliang Lei, Weizhuang Shi, Zejia Zhang, Wei Luo, Weiwei Wan, *Senior Member, IEEE*, Xinxing Chen, *Member, IEEE*, and Jian Huang, *Senior Member, IEEE*

***Abstract*—In this work, we conducted research on deformable object manipulation by robots based on demonstration-enhanced reinforcement learning (RL). We present FADERL (Fuzzy-Augmented Demonstration-Embedded Reinforcement Learning), a novel framework for robotic manipulation of deformable objects that significantly improves reinforcement learning efficiency through synergistic unification of High-Dimensional Takagi-Sugeno-Kang (HTSK) fuzzy systems, Generative Adversarial Behavior Cloning (GABC), and Conditional Policy Learning (CPL). Compared to the Rainbow-DDPG baseline, FADERL achieves 2.01× higher global average reward and reduces standard deviation to 45% while requiring fewer computational resources. To address the high cost of human demonstration collection, we introduce a Nonlinear Model Predictive Control (NMPC)-based data augmentation method that generates high-quality demonstrations at minimal cost. Simulation results demonstrate that NMPC-generated demonstrations enable FADERL to achieve performance comparable to human demonstrations. Physical experiments on fabric manipulation tasks—diagonal folding, central-axis folding, and flattening—achieve success rates of 83.3%, 80.0%, and 96.7% respectively, validating our approach's effectiveness in real-world scenarios. Unlike computationally intensive large-model approaches, FADERL provides a lightweight, task-specific solution with efficient adaptability, making it suitable for practical robotic applications in manufacturing, medical surgery, and service robotics.**

***Note to Practitioners*—This study addresses the challenges of efficiency and cost in robotic manipulation of deformable objects by proposing a lightweight algorithmic framework (FADERL) and a low-cost demonstration data augmentation approach based on NMPC. Traditional methods often rely on extensive manual demonstration data or high computational resources. Our solution integrates fuzzy system optimization, reinforcement learning enhancements, and automated demonstration generation to improve success rates in tasks like folding and flattening while reducing data acquisition costs. Experimental results demonstrate robust performance in both simulation and real-world scenarios. The methodology is applicable to industrial applications (e.g., flexible cable assembly), medical surgery (e.g., soft tissue suturing), and household service robots (e.g., clothing organization). Compared to current large-scale models, the proposed algorithm requires fewer computational resources and supports task-specific customization.**

***Index Terms*—Deformable objects, robotic manipulation, reinforcement learning, demonstration, nonlinear model predictive control.**

## I. Introduction

DEFORMABLE objects play a critical role in numerous key industries and are widely used across various sectors [1]. Their manipulation is a common task in manufacturing [2], [3], medical surgery [4], [5], and service robotics [6], [7], [8]. However, manual handling of deformable objects is labor-intensive, costly, and inefficient. Consequently, robots are often employed to replace human operators for manipulating deformable objects, such as connecting cables on automated assembly lines [9], [10], cutting or suturing soft tissue during medical surgeries [11], and handling fabrics like towels and clothes in home service scenarios [12], [13]. Automating the manipulation of deformable objects with robots can significantly reduce labor costs while improving operational efficiency and precision. Therefore, robotic systems for manipulating deformable objects have attracted considerable attention and research [14].

Currently, most robotic manipulation research focuses on rigid objects. When dealing with deformable objects, robots face challenges like high-dimensional state spaces and complex dynamics [15]. Some researchers build dynamic models for deformable objects, but achieving high-accuracy models is difficult [16], [17]. To avoid this, some researchers turn

Received 7 June 2025; revised 17 September 2025; accepted 26 October 2025. Date of publication 3 November 2025; date of current version 13 November 2025. This article was recommended for publication by Associate Editor C. Zeng and Editor X. Liu upon evaluation of the reviewers' comments. This work was supported in part by Hubei Science and Technology Major Project under Grant 2024BAA007, in part by the National Natural Science Foundation of China under Grant 62333007 and Grant U24A20280, and in part by Hubei Provincial Technology Innovation Program under Grant 2025DJA047. *(Haoyuan Wang and Zihao Dong contributed equally to this work.) (Corresponding authors: Xinxing Chen; Jian Huang.)*

Haoyuan Wang, Tong Zhu, Hongliang Lei, Weizhuang Shi, Zejia Zhang, Xinxing Chen, and Jian Huang are with Hubei Key Laboratory of Brain-Inspired Intelligent Systems, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology (HUST), Wuhan, Hubei 430074, China, and also with the Key Laboratory of Image Processing and Intelligent Control, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology (HUST), Wuhan, Hubei 430074, China (e-mail: why427@hust.edu.cn; zt1021@hust.edu.cn; leihl@hust.edu.cn; swz@hust.edu.cn; zejiazhang@hust.edu.cn; cxx@hust.edu.cn; huang_jan@hust.edu.cn).

Zihao Dong is with China Academy of Aerospace System and Innovation, Beijing 100032, China (e-mail: ddzh130@163.com).

Wei Luo is with the Department of Innovation Center, China Ship Development and Design Center, Wuhan, Hubei 430064, China (e-mail: csddc_weiluo@163.com).

Weiwei Wan is with the Graduate School of Engineering Science, The Osaka University, Toyonaka 560-0043, Japan (e-mail: wan@sys.es.osaka-u.ac.jp).

This article has supplementary downloadable material available at https://doi.org/10.1109/TASE.2025.3627775, provided by the authors.

Digital Object Identifier 10.1109/TASE.2025.3627775

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

to learning-based methods such as reinforcement learning (RL) [18] and imitation learning (IL) [19]. These methods learn control policies from data using learning algorithms, without requiring explicit dynamical modeling. RL involves the agent continuously exploring the action space through trial and error, collecting interaction data with the environment to facilitate learning. Still, in real-world scenarios, the intricacy involved in handling deformable objects frequently results in inefficient learning processes that require extensive time and data, yielding less-than-optimal results. Therefore, it is crucial to set more effective states and action spaces to reduce task complexity based on domain-specific knowledge [20], [21]. With the advancement of deep learning technology, Deep Reinforcement Learning (DRL) is being used to tackle deformable object manipulation problems. Matas et al. [22] trained agents using DRL methods in simulation environments to fold clothes or place them on hangers. Researchers incorporated seven commonly used techniques in the DRL field into the Deep Deterministic Policy Gradient (DDPG) to develop the Rainbow-DDPG algorithm and validated the effectiveness of these techniques through ablation experiments. Additionally, they conducted deformable object manipulation experiments in real scenes through domain randomization. Jangir et al. [23] treated the coordinates of a series of key points on the fabric as the state space, introducing Behavioral Cloning (BC) and Hindsight Experience Replay (HER) to improve the DDPG algorithm for handling tasks involving dynamic manipulation of fabrics by robots. They also studied the impact of key point selection on RL performance. Despite making some progress in learning effective strategies for deformable object manipulation, DRL still faces challenges in terms of learning efficiency due to the inherent complexity of such manipulation tasks, requiring substantial amounts of data and computational resources for training.

Collecting human demonstration data [24] and extracting operational skills using IL algorithms from these demonstrations has also received extensive research attention. Unlike the trial-and-error mechanism of RL, IL completes tasks through observation and imitation of expert behavior. This method has unique advantages in handling tasks that are too complex for RL or where clear rewards are difficult to define [25]. With the continuous development of deep learning technology and hardware infrastructure, recent research has been able to collect a large amount of human demonstration data and utilize deep learning techniques to extract manipulation skills from it [26], [27], [28]. Although extracting manipulation skills from a large amount of human demonstrations can yield decent results, the high labor cost associated with this approach is often challenging and unsustainable. Some studies combine RL and IL, leveraging human demonstrations to enhance the learning efficiency of RL while also benefiting from RL's ability for autonomous exploration [22], [23], [29]. Vodolazskii et al. [29] used the K-means clustering algorithm to categorize human demonstration actions into $M$ classes, testing the feasibility of each type of human demonstration action on the robot and selecting the feasible one with the highest reward as the starting point for agent exploration, thus streamlining the search space of RL algorithms. It is worth mentioning that the Rainbow-DDPG algorithm mentioned earlier [22] and the work by Jangir et al. [23] also incorporate human demonstration data to improve the learning efficiency of RL. Undeniably, the morphological diversity of deformable objects imposes higher requirements on the range of operational scenarios covered by demonstration data. To cover as wide a range of operational scenarios as possible, researchers typically need to collect a large amount of demonstration data. Existing studies [26], [27] use manually collected large-scale demonstration data for training purposes, which inevitably incurs significant human resource costs.

Currently, more and more research tends to adopt vision-language-action (VLA) for robotic manipulation. However, such research often requires significant computational resources and is overqualified when dealing with specific tasks. For example, OpenVLA is a 7B-parameter VLA that was trained on 64 A100 GPUs for 14 days. During inference, it requires 15GB of video memory and runs at approximately 6Hz on an NVIDIA RTX 4090 GPU [30]. The largest model of RT-2 uses 55B parameters, and it is infeasible to directly run such a model on standard desktop machines or on-robot GPUs commonly used for real-time robot control [31]. Even TinyVLA requires 1.3B parameters [32]. Recent studies specifically targeting deformable object manipulation, such as the work by Fu et al. [3] on fabric manipulation with LLMs, Gemini Robotics [33] and GR-3 [34] clothing organization tasks in home service scenarios. These studies have shown promising results but still face substantial challenges: they typically require extensive human demonstration data (often collected through teleoperation across multiple robot platforms), massive computational resources (e.g., clusters of GPUs for training), and significant inference time.

To address the challenges of low learning efficiency in RL and the difficulty in collecting IL demonstration data mentioned above, this article aims to optimize existing RL algorithms from two perspectives: First, to fully and reasonably utilize demonstration data to enhance the learning efficiency of RL on the basis of introducing IL. Second, to propose low-cost methods for collecting demonstration data. Through these, robots can learn manipulation skills with higher efficiency and lower cost, thus operating deformable objects more efficiently in practical applications. Specifically, the main contributions of this article are as follows:

- **FADERL Algorithm**: An RL method enhanced by demonstration to increase the learning efficiency of RL with human demonstration data for training.
- **NMPC-Based Data Augmentation**: A demonstration data augmentation method based on Nonlinear Model Predictive Control (NMPC) to reduce the cost of demonstration data collection.
- **Simulation & Physical Validation**: Validating the feasibility of the proposed methods in simulation and real environments through experiments.

The main research content and relationships are shown in Fig. 1. The specific content arrangement of each section is as follows: Section I is the introduction. Section II addresses the issue of low learning efficiency in RL algorithms by proposing FADERL (Fuzzy-Augmented Demonstration-Embedded

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

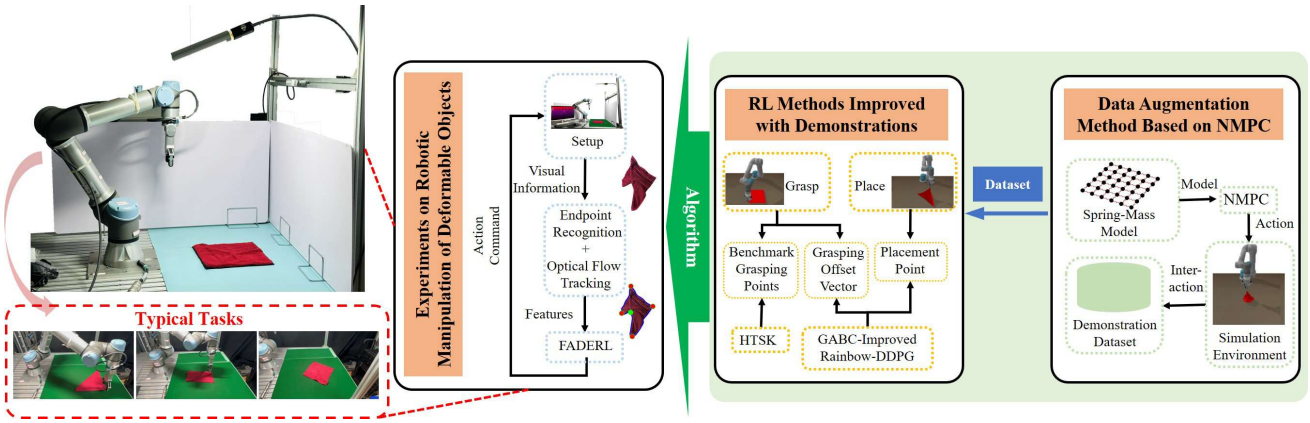



Fig. 1. The diagram of the proposed FADERL algorithm and NMPC-based data augmentation method.

Reinforcement Learning) algorithm that incorporates a High-Dimensional Takagi-Sugeno-Kang (HTSK) fuzzy system, Generative Adversarial Behavior Cloning (GABC) techniques, Rainbow-DDPG, and Conditional Policy Learning (CPL). Section III addresses the issue of high cost of collecting demonstration data by proposing a low-cost data augmentation method based on NMPC. Section IV presents the simulation and physical experiment settings that validate the methods proposed in this article. Section V presents the experiment results. Section VI provides a summary of the entire article and outlooks future work.

## II. Fuzzy-Augmented Demonstration-Embedded Reinforcement Learning

### A. Task Setting

For a square piece of fabric as a representative deformable object, these tasks are designed: folding along the diagonal, folding along the central-axis, and flattening. The details are as followed:

- **Folding along the diagonal:** The specific objective of the operation is to achieve perfect alignment of one pair of diagonal endpoints of the fabric.
- **Folding along the central-axis:** After folding, the two sets of fabric endpoints should coincide, while ensuring that the distance between endpoints on the side of the folding axis remains consistent with before folding.
- **Flattening:** When faced with heavily wrinkled fabric, the robot's task is to flatten it to its maximum area.

In this work, a "demonstration" refers to a complete trajectory of state-action-reward-next state tuples collected during a single task execution. Each demonstration represents a successful or partially successful attempt to complete the target task and serves as structured training data for imitation learning components of our framework.

### B. State, Action, and Reward

*1) State Space:* This article selects the positions of the four corners of the square fabric and the proportion $f_t$ of the fabric's current area to its area when fully flattened as the state representation in these two folding tasks. This results in a 13-dimensional state space, as shown in Fig. 2(a). The symbols defining the state variables for the folding tasks are as follows:

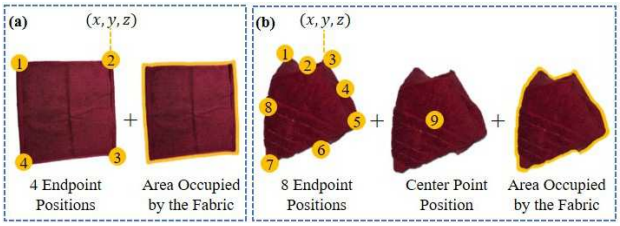



Fig. 2. State spaces for different tasks. (a) Folding Task. (b) Flattening Task.

$$\boldsymbol{s}_t = (\boldsymbol{p}_{1_t}, \boldsymbol{p}_{2_t}, \boldsymbol{p}_{3_t}, \boldsymbol{p}_{4_t}, f_t) \tag{1}$$

where $\boldsymbol{p}_{1_t}, \boldsymbol{p}_{2_t}, \boldsymbol{p}_{3_t}, \boldsymbol{p}_{4_t}$ respectively represent the three-dimensional coordinates of the four corners of the fabric at time $t$. All coordinates and vectors mentioned in this article are described with respect to the base coordinate system of the robot they are associated with.

In the flattening task, the fabric's initial state is heavily wrinkled, making it extremely difficult to detect the right-angle corners of the fabric. We fit the contour of the fabric into an octagon. The coordinates of the eight endpoints, the coordinates of the center point of the fabric contour and the proportion of the fabric's current area to its area when fully flattened are the state representation, as shown in Fig. 2(b). The symbols defining the state variables for the flattening task are as follows:

$$\boldsymbol{s}_t = (\boldsymbol{p}_{1_t}, \boldsymbol{p}_{2_t}, \cdots, \boldsymbol{p}_{8_t}, \boldsymbol{p}_{c_t}, f_t) \tag{2}$$

where $\boldsymbol{p}_{1_t}, \boldsymbol{p}_{2_t}, \cdots, \boldsymbol{p}_{8_t}$ respectively represent the three-dimensional coordinates of the eight fitted endpoints of the fabric at time $t$, and $\boldsymbol{p}_{c_t}$ represents the three-dimensional coordinates of the center point of the fabric contour at time $t$.

*2) Action Space:* This article designs a motion vector $\boldsymbol{\delta}_t$, as illustrated in Fig. 3. The operation process are refined into three key steps: firstly, selecting one endpoint from the state variables as the base grasping point; secondly, generating an offset vector $\boldsymbol{\delta}_t$ to accurately adjust the position of the grasping point; thirdly, determining the coordinates of the placement

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.


Fig. 3. Illustration of offset vectors. Left: grasping the Endpoint Belonging to the Connecting Part. Right: grasping a Point Belonging to the Upper Layer.

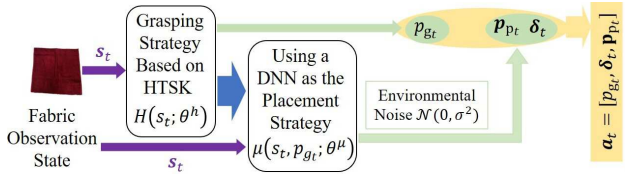



Fig. 4. Illustration of conditional policy learning.

point to guide the robot to complete the entire action from grasping to placing. The representation of the action space is described in (3):

$$\boldsymbol{a}_t = (p_{\mathrm{g}_t}, \boldsymbol{\delta}_t, \boldsymbol{p}_{\mathrm{p}_t}) \tag{3}$$

where $p_{\mathrm{g}_t}$ represents the index of the endpoint selected at time $t$ in the state variables, $\boldsymbol{\delta}_t$ represents the offset vector for time $t$, and $\boldsymbol{p}_{\mathrm{p}_t}$ represents the coordinates of the placement point at time $t$.

*3) State Transition:* The state transition from time $t$ to time $t+1$ is controlled by $\boldsymbol{a}_t$ as expressed in (3). Initially, the robot determines the coordinates of the corresponding endpoint $\boldsymbol{p}_{\mathrm{g}_t}$ in $\boldsymbol{s}_t$ based on $p_{\mathrm{g}_t}$. Subsequently, by combining $\boldsymbol{\delta}_t$, the actual grasping coordinates $\boldsymbol{g}_t = \boldsymbol{p}_{\mathrm{g}_t} + \boldsymbol{\delta}_t$ are calculated, guiding the end effector to execute the grasping action at the $\boldsymbol{g}_t$ position. Finally, the robot moves the end effector to the $\boldsymbol{p}_{\mathrm{p}_t}$ position.

*4) Reward Setting:* For the diagonal folding task, the fabric's target state is for endpoint 1 and 3 to coincide, the distance between endpoint 2 and 3 to equal the diagonal length, and the area to be half of the fully-unfolded area. With side-length $l_s$ of the square fabric, the difference $e_t$ between the fabric state at time $t$ is defined as follows:

$$e_t = \left\|\boldsymbol{p}_{1_t} - \boldsymbol{p}_{3_t}\right\|_2 + \left|\sqrt{2}l_s - \left\|\boldsymbol{p}_{2_t} - \boldsymbol{p}_{4_t}\right\|_2\right| + \left|f_t - 0.5\right|. \tag{4}$$

For the central-axis folding task, the goal is to align endpoint 1 with 2, endpoint 3 with 4, keep the distances between endpoints as $l_s$, and the area half of the fully-unfolded area. The difference $e_t$ is as follows:

$$\begin{aligned} e_t = {} & \left\|\boldsymbol{p}_{1_t} - \boldsymbol{p}_{2_t}\right\|_2 + \left\|\boldsymbol{p}_{3_t} - \boldsymbol{p}_{4_t}\right\|_2 + \left|l_s - \left\|\boldsymbol{p}_{1_t} - \boldsymbol{p}_{3_t}\right\|_2\right| \\ & + \left|l_s - \left\|\boldsymbol{p}_{2_t} - \boldsymbol{p}_{4_t}\right\|_2\right| + \left|f_t - 0.5\right|. \end{aligned} \tag{5}$$

The reward functions for $e_t$ in folding tasks is as follows:

$$r(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} -200e_t + 100, & \textit{done} \\ 3, & \text{not } \textit{done} \text{ and } e_{t-1} - e_t > t_z \\ -3, & \text{not } \textit{done} \text{ and } e_t - e_{t-1} > t_z \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where *done* becomes True when the maximum operations number $t_{\mathrm{m}}$ is reached. $t_z$ is the threshold to measure whether the fabric state has significantly changed. At the end of a round, a decisive reward of $-200e_t + 100$ is given based on the error $e_t$. The greater the error, the lower the decisive reward. The decisive reward is set to 100 when the $e_t$ is 0. If $e_{t-1} - e_t > t_z$, indicating the agent is approaching the target, a positive reward of 3 is given to encourage similar behavior. If $e_t - e_{t-1} > t_z$, indicating the agent deviates from the target, a negative penalty of $-3$ is applied to suppress this behavior.

In the flattening task, the reward function is based on the ratio $f_t$ of the fabric's unfolded area to the fully-unfolded area:

$$r(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} 200f_t - 100, & \textit{done} \\ 3, & \text{not } \textit{done} \text{ and } f_t - f_{t-1} > t_z \\ -3, & \text{not } \textit{done} \text{ and } f_{t-1} - f_t > t_z \\ 0, & \text{otherwise} \end{cases}. \tag{7}$$

The criteria are similar to those in (6). However, the reward is allocated based on the fabric's flattening degree $f_t$. Specifically, a larger $f_t$ indicates a higher degree of fabric flatness, and thus the agent receives a correspondingly increased reward, reflecting a proportional relationship.

The design of reward functions was guided by three key criteria. First, alignment with task objectives: For folding tasks, the reward functions directly quantify the deviation from target states (e.g., endpoint alignment, area ratio) via error $e_t$, ensuring the agent learns to minimize such deviations. For the flattening task, the reward is based on the area ratio $f_t$, which directly reflects the task's goal of maximizing fabric spread. Second, incremental guidance: By comparing $e_t$ (or $f_t$) with its previous value, the functions provide immediate feedback ($\pm 3$ rewards) to encourage progress toward the target and suppress regressions, accelerating the learning of effective strategies. Third, terminal evaluation: The decisive reward at the end of a round (e.g., $-200e_t + 100$) ensures the agent prioritizes achieving the final target state, balancing short-term progress with long-term task completion.

### C. FADERL Algorithm

The FADERL algorithm is detailed in Algorithm 1.

*1) Algorithm for Selecting Benchmark Grasping Points Based on the HTSK Fuzzy System:* In this article, choosing an appropriate final placement point $\boldsymbol{p}_{\mathrm{p}_t}$ enables the selection of the reference point $p_{\mathrm{g}_t}$ to advance the task. Thus, the selection of $p_{\mathrm{g}_t}$ relates more to the application range of fuzzy sets. We use state-action pairs $(\boldsymbol{s}_t, \boldsymbol{a}_t)$, with state $\boldsymbol{s}_t$ as input and $p_{\mathrm{g}_t}$ as output, to construct an HTSK fuzzy system $H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$ ($\boldsymbol{\theta}^h$ are its parameters), where $p_{\mathrm{g}_t} = H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$. Training data for the reference grasping point selection strategy come from the human demonstration dataset $\mathcal{D}_{\mathrm{demo}}$, supplemented by task-progress-related data during environment interaction, forming $\mathcal{D}_{\mathrm{grasp}} = (\boldsymbol{s}_n, p_{\mathrm{g}_n})_{n=1}^N$. Here, $N$ is the dataset size, $\boldsymbol{s}_n = (s_1^{(n)}, s_2^{(n)}, \ldots, s_M^{(n)})$ are $M$-dimensional state variables, $p_{\mathrm{g}_n} \in (1, 2, \cdots, k)$ is the reference grasping point index (also the label), and $k$ is the number of candidate points. The system learns the mapping between $\boldsymbol{s}_n$ and $p_{\mathrm{g}_n}$ to predict grasping points. HTSK's main improvement over TSK lies in addressing data-dimension saturation. In TSK, the softmax

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

**Algorithm 1** FADERL

**Require:** Demonstration Dataset $\mathcal{D}_{\text{demo}}$, Total Number of Rounds $M$, Number of Pre-Training Rounds $M_p$, Maximum Number of Operations per Round $t_{\text{m}}$, Number of Strategy Updates per Interaction $t_{\text{n}}$, Batch Size $N$, Random Environmental Noise $\mathcal{N}$.

**Ensure:** Trained HTSK Fuzzy System $H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$, Critic Network $Q(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^q)$ and Offset Vector & Placement Point Selection Strategy Network $\mu(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^\mu)$.

Extract$(\boldsymbol{s}_i,\boldsymbol{a}_i)$ from $\mathcal{D}_{\text{demo}}$to form $\mathcal{D}_{\text{grasp}}$. Add $\mathcal{D}_{\text{demo}}$ to the replay buffer R. Initialize $H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$, $Q(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^q)$, $\mu(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^\mu)$, $Q_1'(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^{q'})$, $Q_2'(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^{q'})$, $\mu'(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^{\mu'})$.
**while** $H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$has not converged **do**
  Use $\mathcal{D}_{\text{grasp}}$ as the training dataset, and update the parameters $\boldsymbol{\theta}^h$ of $H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$ according to the method described in Section II-B.1).
**end while**
**for** $e=1,M$ **do**
  Initialize the environment and receive the initial observation state $\boldsymbol{s}_1$.
  **for** $t=1,t_{\text{m}}$ **do**
    Referring to Fig. 4, generate $\boldsymbol{a}_t$ by combining $H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$, $\mu(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^\mu)$, and noise $\mathcal{N}$.
    Execute action $\boldsymbol{a}_t$to interact with the environment, and add $(\boldsymbol{s}_t,\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1})$ to the replay buffer R.
    **for** $b=1,t_{\text{n}}$ **do**
      **if** $e\le M_p$ **then**
        Set $\lambda_{\text{diff}}$ to 1, and sample $N$ data points $(\boldsymbol{s}_i,\boldsymbol{a}_i,r_i,\boldsymbol{s}_{i+1})$ from $\mathcal{D}_{\text{demo}}$.
      **else**
        Set $\lambda_{\text{diff}}$ to 0, and sample $N$ data points $(\boldsymbol{s}_i,\boldsymbol{a}_i,r_i,\boldsymbol{s}_{i+1})$ from R.
      **end if**
      Update $Q(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^q)$ by minimizing the loss $L_{\text{Critic}}$ defined in (12).
      Update $\mu(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^\mu)$ using the method in Rainbow-DDPG.
      If $\boldsymbol{a}_i$significantly advances the task, add $(\boldsymbol{s}_i,\boldsymbol{a}_i)$ to $\mathcal{D}_{\text{grasp}}$. If there are more than 50 new data points added to $\mathcal{D}_{\text{grasp}}$, retrain $H(\boldsymbol{s}_t;\boldsymbol{\theta}^h)$.
    **end for**
    Soft-update the parameters of the target networks $Q'(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^{q'})$ and $\mu'(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^{\mu'})$.
  **end for**
**end for**

function calculates the normalized firing level $\overline{\omega}_r$of each fuzzy rule as:

$$\overline{\omega}_r=\frac{\exp(H_r)}{\sum_{r=1}^{R}\exp(H_r)}. \tag{8}$$

As the input data dimension $M$ increases, $H_r$ decreases, saturating the softmax function and reducing the classification performance of TSK [35]. HTSK replaces $\overline{\omega}_r$ with its mean $H_r^*$ to better handle high-dimensional data [36]. In the sixth and seventh layers, the final $p_{\text{g}_t}$ is selected as output based on the softmax function and probability distribution.

We use k-means clustering to initialize $c_{r,m}$ (parameters of the Gaussian membership function), the cross-entropy loss function, and the Adam optimizer (learning rate 0.04, batch size 64, weight decay 1e-8) for training.

*2) GABC-Improved Rainbow-DDPG Algorithm:* In Rainbow-DDPG, BC usually adds $L_{\text{bc}}$ to the Actor network's loss function, but training the Critic network for accurate Q-values is time-consuming, reducing $L_{\text{bc}}$'s early-stage effectiveness and the chance of sampling demonstration data. We propose GABC to improve Rainbow-DDPG. The Actor network $\mu(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^\mu)$ combines noise $\mathcal{N}(0,\sigma^2)$ to output $(\boldsymbol{\delta}_t,\boldsymbol{p}_{\text{p}_t})$, as follows:

$$(\boldsymbol{\delta}_t,\boldsymbol{p}_{\text{p}_t})=\mu(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^\mu)+\mathcal{N}(0,\sigma^2) \tag{9}$$

where $\boldsymbol{\theta}^\mu$ represents its parameters. The Critic network $Q(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^q)$ evaluates action quality. This study denotes $(\boldsymbol{\delta}_i,\boldsymbol{p}_{\text{p}_i})$ as $\boldsymbol{b}_i$, and sets $\boldsymbol{w}_i=(p_{\text{g}_i},\mu(\boldsymbol{s}_i,p_{\text{g}_i};\boldsymbol{\theta}^\mu))$. Considering the relationship between $p_{\text{g}_i}$, $\boldsymbol{\delta}_t$ and $\boldsymbol{p}_{\text{p}_t}$, $L_{\text{bc}}$ can be redefined as follows:

$$L_{\text{bc}}=\begin{cases}(\mu(\boldsymbol{s}_t,p_{\text{g}_t};\boldsymbol{\theta}^\mu)-\boldsymbol{b}_i)^2, & Q(\boldsymbol{s}_i,\boldsymbol{a}_i;\boldsymbol{\theta}^q)>Q(\boldsymbol{s}_i,\boldsymbol{w}_i;\boldsymbol{\theta}^q)\\ & \text{and } (\boldsymbol{s}_i,\boldsymbol{a}_i)\in\mathcal{D}_{\text{demo}}\\ 0, & \text{otherwise}\end{cases}. \tag{10}$$

GABC introduces $Q_{\text{diff}}$ to the Critic network's loss function. Assuming $(\boldsymbol{s}_i,\boldsymbol{a}_i)\in\mathcal{D}_{\text{demo}}$, $Q_{\text{diff}}$ guides the training of the Critic network by measuring the difference between the $Q$-values output by the current Critic network for the actions $\boldsymbol{a}_i$ in the human demonstration data and the $Q$-values output for the actions $\boldsymbol{w}_i=(p_{\text{g}_i},\mu(\boldsymbol{s}_i,p_{\text{g}_i};\boldsymbol{\theta}^\mu))$ by the current Actor network, under the same state $\boldsymbol{s}_i$. Specifically, this article sets a pre-training stage, as shown in (11):

$$Q_{\text{diff}}=\max(0,100-(Q(\boldsymbol{s}_i,\boldsymbol{a}_i;\boldsymbol{\theta}^q)-Q(\boldsymbol{s}_i,\boldsymbol{w}_i;\boldsymbol{\theta}^q))). \tag{11}$$

After pre-training, the risk of the Actor network getting stuck in local optima due to $Q_{\text{diff}}$ exists. The loss function $L_{\text{Critic}}$ for the improved Critic network $Q(\boldsymbol{s}_t,\boldsymbol{a}_t;\boldsymbol{\theta}^q)$ is defined as follows:

$$L_{\text{Critic}}=\lambda_{1\text{step}}L_{1\text{s}}+\lambda_{n\text{step}}L_{n\text{s}}+\lambda_{\text{diff}}Q_{\text{diff}} \tag{12}$$

where $\lambda_{1\text{step}}$ and $\lambda_{n\text{step}}$ are the weights of the 1-step and n-step TD loss functions, respectively. $\lambda_{\text{diff}}$ is the weight of $Q_{\text{diff}}$, set to 1 during the pre-training phase and 0 in subsequent phases. $L_{1\text{s}}$ and $L_{n\text{s}}$ are similar to the 1-step and n-step TD loss functions [22], and the target functions $y_{1\text{s}}$ and $y_{n\text{s}}$ can be referenced from TD3 model [38]. Here, we define $Q_1'$, $Q_2'$, and $\mu'$ as two target Critic networks and one target Actor network, with $\boldsymbol{w}'_{i+k}=(H(\boldsymbol{s}_{i+k};\boldsymbol{\theta}^h),\mu'(\boldsymbol{s}_{i+k},H(\boldsymbol{s}_{i+k};\boldsymbol{\theta}^h);\boldsymbol{\theta}^{\mu'}))$, $k=1,2,\cdots,n$, where $n$ is the step length of the n-step TD loss function, $N$ is the batch size, $r_i$ is the reward, and $\gamma$ is the discount factor. Consequently, $L_{1\text{s}}$ and $L_{n\text{s}}$ are defined as follows:

$$L_{1\text{s}}=\frac{1}{N}\sum_{i=1}^{N}(Q(\boldsymbol{s}_i,\boldsymbol{a}_i;\boldsymbol{\theta}^q)-y_{1\text{s}})^2, \tag{13}$$
$$y_{1\text{s}}=r_i+\gamma\min_{j=1,2}Q_j'(\boldsymbol{s}_{i+1},\boldsymbol{w}'_{i+1};\boldsymbol{\theta}^{q'}),$$

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

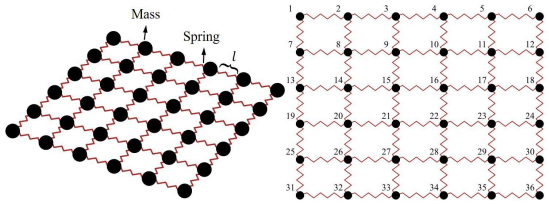



Fig. 5. Illustration of the Spring-Mass model and node numbering.

$$L_{ns} = \frac{1}{N}\sum_{i=1}^{N}(Q(\boldsymbol{s}_i, \boldsymbol{a}_i; \boldsymbol{\theta}^q) - y_{ns})^2,$$
$$y_{ns} = \sum_{t=0}^{n-1}\gamma^t r_{i+t+1} + \gamma^n \min_{j=1,2} Q'_j(\boldsymbol{s}_{i+n}, \boldsymbol{w}'_{i+n}; \boldsymbol{\theta}^{q'}). \quad (14)$$

$Q_{\text{diff}}$ makes the Critic network assign higher Q-values to human-demonstration actions, enhancing $L_{\text{bc}}$'s effect and accelerating Actor-network training. Since the Actor network depends on the Critic network's Q-values, a more accurate Critic network further speeds up Actor-network training.

*3) CPL Learning Method:* This article adopts the CPL training method illustrated in Fig. 4. At each state $\boldsymbol{s}_t$, CPL first utilizes the $H(\boldsymbol{s}_t; \boldsymbol{\theta}^h)$ to select $p_{\text{g}_t}$, and then, based on $p_{\text{g}_t}$, selects $\boldsymbol{\delta}_t$ and $\boldsymbol{p}_{\text{p}_t}$ through $\mu(\boldsymbol{s}_t, p_{\text{g}_t}; \boldsymbol{\theta}^\mu)$. CPL often faces the loss-allocation challenge [37]. It's hard to determine if high rewards for an action come from the grasping-point selection strategy $H(\boldsymbol{s}_t; \boldsymbol{\theta}^h)$ or the offset-vector and placement-point selection strategy $\mu(\boldsymbol{s}_t, p_{\text{g}_t}; \boldsymbol{\theta}^\mu)$. To address this, we adopt the following training approach: First, extract each state-action pair $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ from $\mathcal{D}_{\text{demo}}$ to form the initial training dataset $\mathcal{D}_{\text{grasp}}$. Then, use $\mathcal{D}_{\text{grasp}}$ to train $H(\boldsymbol{s}_t; \boldsymbol{\theta}^h)$ for $\boldsymbol{\theta}^h$. Next, with $\boldsymbol{\theta}^h$ fixed, based on $H(\boldsymbol{s}_t; \boldsymbol{\theta}^h)$- selected $p_{\text{g}_t}$, train $\mu(\boldsymbol{s}_t, p_{\text{g}_t}; \boldsymbol{\theta}^\mu)$ using the improved Rainbow-DDPG algorithm with GABC enhancement for $\boldsymbol{\theta}^\mu$. During this process, continuously collect data to supplement $\mathcal{D}_{\text{grasp}}$ and train $\boldsymbol{\theta}^h$ of $H(\boldsymbol{s}_t; \boldsymbol{\theta}^h)$. Specifically, when action $\boldsymbol{a}_t$ at time $t$ significantly progresses the task (in folding tasks, $e_{t-1} - e_t > t_z$; in flattening tasks, $f_t - f_{t-1} > t_z$), $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is added to $\mathcal{D}_{\text{grasp}}$. After getting new data, use $\mathcal{D}_{\text{grasp}}$ to retrain $H(\boldsymbol{s}_t; \boldsymbol{\theta}^h)$ for new $\boldsymbol{\theta}^h$.

## III. Data Augmentation Method Based on NMPC

### A. Establishment and Analysis of the Spring-Mass Model

We introduce a low-cost demonstration data augmentation method based on NMPC. An NMPC problem is built upon the spring-mass model to obtain optimal control strategies for specific tasks. However, extracting the states of all particles in complex deformable object states (e.g., highly wrinkled fabric) poses challenges in real-world scenarios. Thus, NMPC aims to collect demonstration data in simulation. The spring-mass model adopted here, as illustrated in Fig. 5, is a system of particles connected by springs, with the fabric's mass evenly distributed. It consists of $N_{\text{sp}}$ particles $\mathcal{L} = (1, 2, \cdots, N_{\text{sp}})$ and $M$ springs $\mathcal{S}$. The neighbor set $\mathcal{N}_i$ of particle $i$, i.e., particles connected to it by springs, is assumed fixed.

For a square fabric, $N_{sp} = n^2$ ($n$ is the number of particles per side). Each particle has mass $m_i = m, \forall i$, and its position at time $t$ is $\boldsymbol{x}_t^i = (x_t^i, y_t^i, z_t^i)$. Spring stiffness $k$ depends on material properties, and natural length $l$ on the fabric's initial state; we assume all springs have the same $k$ and $l$. The neighbor set $\mathcal{N}_i = \{j | (i, j) \in \mathcal{S}\}$. Particles are numbered row-by-row. Fig. 5 illustrates the case when $n = 6$. For any two spring-connected particles $i$ and $j$, the spring force $\boldsymbol{s}_t^{i,j}$ at time $t$ is:

$$\boldsymbol{s}_t^{i,j} = k(l_t^{i,j} - l)\frac{\boldsymbol{x}_t^i - \boldsymbol{x}_t^j}{l_t^{i,j}} \quad (15)$$

where $l_t^{i,j} = \left\|\boldsymbol{x}_t^i - \boldsymbol{x}_t^j\right\|$. Each particle is also influenced by gravity $\boldsymbol{G} = m\boldsymbol{g}$, external force $\boldsymbol{u}_t^i$, and damping force $\boldsymbol{d}_t^{i,j} = -c(\boldsymbol{v}_t^i - \boldsymbol{v}_t^j)$, with $c$ as the damping coefficient and $\boldsymbol{v}_t^i$ the velocity vector. The total force $\boldsymbol{F}_t^i$ on particle $i$ at time $t$ is:

$$\boldsymbol{F}_t^i = \sum_{j\in\mathcal{N}_i}\boldsymbol{s}_t^{i,j} + \boldsymbol{G} + \boldsymbol{u}_t^i + \sum_{j\in\mathcal{N}_i}\boldsymbol{d}_t^{i,j}. \quad (16)$$

The acceleration $\boldsymbol{a}_t^i = \boldsymbol{F}_t^i/m$, and the velocity updates as $\boldsymbol{v}_{t+\Delta t}^i = \boldsymbol{v}_t^i + \Delta t \cdot \boldsymbol{a}_t^i$. With a small time step $\Delta t$ and a large positive $c$ for stability, the particle position update formula is

$$\boldsymbol{x}_{t+1}^i = \boldsymbol{x}_t^i + \Delta t \cdot \boldsymbol{v}_t^i + \frac{1}{2}\Delta t^2 \cdot \boldsymbol{a}_t^i. \quad (17)$$

### B. Data Augmentation Based on NMPC

NMPC aims to find an optimal control sequence $\boldsymbol{U}_{list}^* = (\boldsymbol{U}_t^*, \boldsymbol{U}_{t+1}^*, \cdots, \boldsymbol{U}_{t+H_p-1}^*)$ within the prediction horizon $H_p$ to minimize the objective function $J$. The system's state $\boldsymbol{X}_t = (\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \cdots, \boldsymbol{x}_t^{N_{sp}})$ consists of particle coordinates, and control input $\boldsymbol{U}_t = (\boldsymbol{u}_t^1, \boldsymbol{u}_t^2, \cdots, \boldsymbol{u}_t^{N_{sp}})$ is the external force on each particle. The spring-mass model's state-transition equation is as follows:

$$\begin{aligned}\boldsymbol{X}_{t+1} &= \boldsymbol{f}(\boldsymbol{X}_t, \boldsymbol{U}_t)\\ &= \boldsymbol{X}_t + \Delta t \cdot \boldsymbol{V}_t + \frac{1}{2\cdot m}\Delta t^2 \cdot (\boldsymbol{S}_t + \boldsymbol{G} + \boldsymbol{U}_t + \boldsymbol{D}_t)\end{aligned} \quad (18)$$

with $\boldsymbol{V}_t$, $\boldsymbol{S}_t$, $\boldsymbol{G}$, $\boldsymbol{D}_t$, and $m$ representing velocity, spring force, gravity, damping force, and particle mass, respectively. For a given task objective, the loss function $L(\boldsymbol{X}_t)$ is:

$$L(\boldsymbol{X}_t) = \sum_{i,j\in\mathcal{P}} w_{i,j}(l_t^{i,j} - l_{\text{ref}}^{i,j})^2 \quad (19)$$

where $\mathcal{P}$ is the set of all pairs of particles to be considered, $l_{\text{ref}}^{i,j}$ is the desired distance between particles $i$ and $j$ in the target state $\boldsymbol{X}_{\text{ref}}$, $w_{i,j}$ is the weight factor used to adjust the relative importance of the differences in distances between different pairs of particles. Taking the particle ordering in Fig. 5 as an example, the target state $\boldsymbol{X}_{\text{ref}}$ is redefined as follows:

1. Diagonal folding: For any pair of particles $i$ and $j$ symmetrically positioned about the specified diagonal:

$$l_t^{i,j} = 0, \text{ such as } l_t^{1,36} = l_t^{8,29} = \ldots = 0. \quad (20)$$

2. Central-axis folding: For any pair of particles $i$ and $j$ symmetrically positioned about the specified central-axis:

$$l_t^{i,j} = 0, \text{ such as } l_t^{1,6} = l_t^{8,11} = \ldots = 0. \quad (21)$$

3. Flattening: For the particle pairs $(i, j)$ and $(a, b)$ at the two diagonals endpoints of the cloth:

$$l_t^{i,j} = l_t^{a,b} = \sqrt{2}l_s, \text{ such as } l_t^{1,36} = l_t^{6,31} = \sqrt{2}l_s \quad (22)$$

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

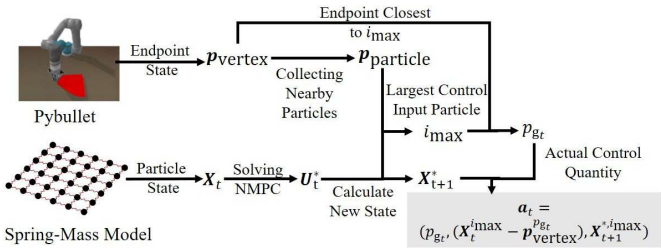



Fig. 6. Process for generating the robot's motion space based on NMPC.

where $l_s$ is the side length of the cloth fully flattened.

NMPC minimizes cumulative loss in the prediction horizon and predicts future states using the Interior Point OPTimizer (Ipopt) [39], which can be implemented by solving the following optimization problem in (23):

$$\begin{aligned}
&\underset{\boldsymbol{U}_{list}}{\text{minimize}} J(\boldsymbol{X}_t, \boldsymbol{U}_{list}) = \sum_{k=t+1}^{t+H_p} L(\boldsymbol{X}_{k|t}) \\
&= \sum_{k=t+1}^{t+H_p} \sum_{i,j\in\mathcal{P}} w_{i,j}(l_{k|t}^{i,j} - l_{\text{ref}}^{i,j})^2 \\
&\text{subject to} \\
&\quad \boldsymbol{X}_{k+1|t} = \boldsymbol{f}(\boldsymbol{X}_{k|t}, \boldsymbol{U}_k) \\
&\quad k = t, \ldots, t + H_p - 1 \\
&\quad \boldsymbol{X}_{t|t} = \boldsymbol{X}_t \\
&\quad -10N \le u_{k,x}^i, u_{k,y}^i, u_{k,z}^i \le 10N, \\
&\quad (i = 1, \ldots, N,\ k = t, \ldots, t + H_p - 1), \\
&\quad \boldsymbol{x}_{k|t}^i \in \mathcal{X}_{\text{workspace}}, \\
&\quad (i = 1, \ldots, N,\ k = t + 1, \ldots, t + H_p)
\end{aligned} \tag{23}$$

where $\boldsymbol{X}_{k|t}$ and $\boldsymbol{X}_{k+1|t}$ represent the states at time steps $k$ and $k+1$ predicted based on the current state $\boldsymbol{X}_t$ and a series of control inputs $\boldsymbol{U}_{list}$. Specifically, when $k = t$, $\boldsymbol{X}_{k|t} = \boldsymbol{X}_{t|t} = \boldsymbol{X}_t$. $J(\boldsymbol{X}_t, \boldsymbol{U}_{list})$ represents the cumulative loss function over the entire prediction horizon $H_p$. The term $l_{k|t}^{i,j}$ denotes the distance between the $i$-th and $j$-th particles calculated based on $\boldsymbol{X}_{k|t}$ at the $k$-th time step. $\mathcal{X}_{\text{workspace}}$ denotes the set of state constraints in the robot's workspace. By solving the optimization problem described above, we can obtain the optimal control sequence $\boldsymbol{U}_{list}^*$.

The optimal control sequence $\boldsymbol{U}_t^*$ from NMPC is mapped to the robot's action space. The specific process is illustrated in Fig. 6. First, apply $\boldsymbol{U}_t^*$ to predict $\boldsymbol{X}_{t+1}^*$, as follows:

$$\boldsymbol{X}_{t+1}^* = \boldsymbol{f}(\boldsymbol{X}_t, \boldsymbol{U}_t^*). \tag{24}$$

Then, in the PyBullet environment, extract endpoint coordinates $\boldsymbol{P}_{\text{vertex}}$, identify neighboring particles to form $\boldsymbol{P}_{\text{particle}}$. Analyze $\boldsymbol{U}_t^*$ to find the maximum force $\boldsymbol{u}_t^{*,i_{\max}}$, determine the reference grasping point $p_{g_t}$, calculate the offset vector $\boldsymbol{\delta}_t$, and set the placement point $\boldsymbol{p}_{p_t}$ based on $\boldsymbol{X}_{t+1}^*$, resulting in:

$$\boldsymbol{a}_t = (p_{g_t}, (\boldsymbol{x}_t^{i_{\max}} - \boldsymbol{p}_{\text{vertex}}^{p_{g_t}}), \boldsymbol{x}_{t+1}^{*,i_{\max}}). \tag{25}$$

Additionally, due to potential simulation errors in the spring-mass particle model, we utilize the PyBullet simulation environment to correct the errors in the spring-mass particle model at each time step, as illustrated in Fig. S1. After obtaining $\boldsymbol{a}_t$ from NMPC as in (25), it is executed in PyBullet to obtain the new state of the cloth, which is then used as the initial state $\boldsymbol{X}_{t+1}$ for the next control cycle, thereby achieving precise updating of the cloth state. We use $dn$ as the termination condition ($|e_t - e_{t-1}| \le 0.01$ for folding, $|f_t - f_{t-1}| \le 0.01$ for flattening) and collect high-reward episodes as the $\mathcal{D}_{\text{NMPC}}$ dataset, as shown in Algorithm 2.

**Algorithm 2** Data Augmentation Based on NMPC

**Require:** System model $\boldsymbol{f}(\boldsymbol{X}_t, \boldsymbol{U}_t)$, initial state $\boldsymbol{X}_0$, prediction horizon $H_p$, cost function $J(\boldsymbol{X}_t, \boldsymbol{U}_{list})$, reward threshold $r_{\text{ts}}$ for each task.

**Ensure:** NMPC demonstration dataset $\mathcal{D}_{\text{NMPC}}$.

```
Initialize temporary dataset D_temp, set dn to False.
while sufficient demonstration data has not been collected do
    Formulate the NMPC optimization problem according to (23).
    while not dn do
        Solve the optimization problem using Ipopt to obtain the control sequence U*_list = (U*_t, U*_{t+1}, ..., U*_{t+H_p-1}).
        Extract U*_t and generate the action vector a_t according to (25).
        Apply a_t to the PyBullet simulation environment to control the robot and update the fabric state.
        Retrieve the new fabric state X_{t+1}, as well as the state vector s_{t+1}, reward r_t, and done flag dn from the PyBullet simulation environment.
        Store (s_t, a_t, r_t, s_{t+1}) in D_temp.
        Set X_{t+1} as the initial state for the next control cycle.
    end while
    if r_t ≥ r_ts then
        Add D_temp to D_NMPC.
    end if
    Clear D_temp and reset the simulation environment for the next control task.
end while
```

## IV. Experimental Setup

The experiments were conducted on a high-performance server equipped with 48GB of RAM, an Intel E5-2660 v4 processor, and an NVIDIA RTX 3090 GPU. The deformable object manipulation system in this article consists of sensor subsystems, decision and control subsystems, and robot motion subsystems, as illustrated in Fig. S2.

### A. Simulation Experiment Settings of the Improved FADERL Algorithm With Human Demonstrations

A complex deformable object manipulation simulation environment was established using PyBullet. The process of the robot grasping the cloth was simulated by establishing precise anchor connections between the corresponding nodes of the cloth model and the robot end-effector. As shown in Fig. S3,

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

TABLE I
HUMAN DEMONSTRATION DATASET

| Task | Average Reward | Reward Standard Deviation | Average Steps |
|---|---|---|---|
| Diagonal Folding | 93.662 | 3.105 | 1.000 |
| Central-Axis Folding | 87.741 | 3.624 | 2.868 |
| Flattening | 87.688 | 5.572 | 8.291 |

in simulation, humans guide the robot to perform grasping and placing actions by clicking on the grasp point $\boldsymbol{p}_{\mathrm{gr}_t}$ and the placement point $\boldsymbol{p}_{\mathrm{p}_t}$ with a mouse, respectively. During this process, we identify the endpoint closest to $\boldsymbol{p}_{\mathrm{gr}_t}$ in state $\boldsymbol{s}_t$, denote its coordinates as $\boldsymbol{p}_{\mathrm{g}_t}$, and its index as $p_{\mathrm{g}_t}$, and calculate the offset vector as $\boldsymbol{\delta}_t = \boldsymbol{p}_{\mathrm{gr}_t} - \boldsymbol{p}_{\mathrm{g}_t}$. Subsequently, we obtain the action $\boldsymbol{a}_t = (p_{\mathrm{g}_t}, \boldsymbol{\delta}_t, \boldsymbol{p}_{\mathrm{p}_t})$, and organize it along with state, reward, and other information into a tuple $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$. At the end of each demonstration, all tuples of the round are stored in a dedicated dataset $\mathcal{D}_{\mathrm{demo}}$ to assist in training the DRL. During data collection, the end condition for rounds of both folding tasks is $e_t \leq 0.1$, and for flattening tasks, the end condition for rounds is $f_t \geq 0.9$. Our demonstration data collection process captures not only the visual state of the deformable object but also the precise action parameters (grasping point index, offset vector, and placement point) and corresponding rewards. This structured representation enables direct utilization in reinforcement learning algorithms without requiring additional processing steps to extract meaningful actions from video data. Table I analyzes results of human demonstration datasets for three different tasks. For the diagonal folding task, the demonstrators achieve the highest average reward, the most stable performance, and the fewest rounds to complete the task. While for the central-axis folding task, the quality of task execution remains relatively high, consistency and efficiency have slightly decreased compared to diagonal folding. In the flattening task, although the average reward is comparable to that of central-axis folding, the standard deviation of the reward significantly increased, and the number of steps required to complete the task surged, indicating the high complexity of the flattening task.

We collected 100 rounds of demonstrations per task through human interaction in simulation, as $\mathcal{D}_{\mathrm{demo}}$. We defined FADERL as Setting 1, and comparative algorithms include:

*1) Setting 2: Rainbow-DDPG:* Rainbow-DDPG uses the same neural network structure for $\mu(\boldsymbol{s}_t, p_{\mathrm{g}_t}; \boldsymbol{\theta}^{\mu})$, $Q(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}^{q})$, and target networks $\mu'$, $Q'_1$ and $Q'_2$. These networks consist of an input layer, 50 hidden fully connected layers (16 neurons each), and an output layer. The loss function $\mu(\boldsymbol{s}_t, p_{\mathrm{g}_t}; \boldsymbol{\theta}^{\mu})$ is defined as (10). The original Rainbow-DDPG employs a neural network instead of HTSK for $H(\boldsymbol{s}_t; \boldsymbol{\theta}^{h})$. During the training process, a cross-entropy loss function is used as the BC loss function $L_{\mathrm{bc}}$ for $H(\boldsymbol{s}_t; \boldsymbol{\theta}^{h})$, defined as follows:

$$L_{\mathrm{bc}} = \begin{cases} -\sum_{p=1}^{k} \delta(p, p_{\mathrm{g}_i}) \log(H_p), & Q(\boldsymbol{s}_i, \boldsymbol{a}_i) \in \mathcal{D}_{\mathrm{grasp}} \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

where $k$ is the number of candidate endpoints. $\delta(p, p_{\mathrm{g}_i})$ is the Kronecker delta function, which equals 1 when $p = p_{\mathrm{g}_i}$, otherwise it's 0. $H_p$ is the probability corresponding to the $p$-th endpoint in the output probability distribution $H(\boldsymbol{s}_i; \boldsymbol{\theta}^{h})$.It's assumed that the grasp point selected from $\mathcal{D}_{\mathrm{grasp}}$ is consistently superior to $H(\boldsymbol{s}_t; \boldsymbol{\theta}^{h})$. The cross-entropy loss function is applied whenever the sampled data $(\boldsymbol{s}_i, \boldsymbol{a}_i)$ comes from $\mathcal{D}_{\mathrm{grasp}}$; otherwise, $L_{\mathrm{bc}}$ is set to 0. Note that in Rainbow-DDPG, the training strategy does not include CPL. This means $H(\boldsymbol{s}_t; \boldsymbol{\theta}^{h})$ is trained in the same way as the current Actor, and its output is not used as part of the input for $\mu(\boldsymbol{s}_t, p_{\mathrm{g}_t}; \boldsymbol{\theta}^{\mu})$.

*2) Setting 3: Rainbow-DDPG + GABC:* Integrates GABC into Rainbow-DDPG.

*3) Setting 4: Rainbow-DDPG + CPL + HTSK:* Incorporates CPL and uses HTSK for grasping-point selection.

*4) Setting 5: Rainbow-DDPG + CPL + Uniform:* Combines CPL with a uniform grasping-point selection strategy.

*5) Setting 6: Rainbow-DDPG + CPL + Random:* Integrates CPL with a random grasping-point selection strategy.

*6) Setting 7: Rainbow-DDPG + GABC + CPL + Uniform:* Integrates GABC and CPL with a uniform strategy.

*7) Setting 8: Rainbow-DDPG + GABC + CPL + Random:* Integrates GABC and CPL with a random strategy.

We designed two modes, simple and challenging, for each of the three tasks. In the simple mode, the maximum number of operations for each task is set to $t_{\mathrm{m}}$; while in the challenging mode, this limit is halved to $\frac{t_{\mathrm{m}}}{2}$ ($t_{\mathrm{m}}$ is 2 for fold along diagonal tasks, 4 for fold along central-axis tasks, and 10 for flattening tasks). Additionally, to investigate the specific impact of the amount of human demonstration data, the models were trained with human demonstration data from 5, 20, and 100 rounds, respectively.

During the training phase, we first performed 20 rounds of pre-training to optimize the Critic network using GABC technique. Subsequently, training proceeded to the regular phase, which consisted of 30 training epochs, with each epoch comprising 20 rounds of training and a batch size ($N$) of 64. After executing a set of actions (grasping and manipulation), the policy was updated $t_{\mathrm{n}}$ times. The single interaction update counts ($N$) for policies in fold along diagonal tasks, fold along central-axis tasks, and flattening tasks were 80, 40, and 20, respectively. At the end of each training epoch, a testing phase comprising 10 rounds was implemented. Specifically, the initial policy's performance was evaluated upon completion of policy initialization, with testing frequency increased during the pre-training phase (testing every 5 rounds of training). Across the study, 35 testing epochs were ultimately completed, with all experiments replicated using three different random seeds to ensure statistical reliability. At the end of the $t$-th testing epoch ($t = 0, \ldots, 34$), for each seed $i$ ($i = 1, 2, 3$), we recorded the total reward $R_j^{i,t}$, $j = 1, \ldots, 10$ obtained by the agent in a single round. Subsequently, the average reward $R_{\mathrm{avg}}^{i,t}$ for each seed was computed across the ten testing instances. We define several key performance metrics:

1. $R_{\mathrm{avg}}^{t}$: The average of all $R_{\mathrm{avg}}^{i,t}$ values within the $t$-th testing epoch. All reward curves presented in this article are plotted based on $R_{\mathrm{avg}}^{i,t}$.
2. $R_{\mathrm{avg}}$: The average of all $R_t^{avg}$ values for $t = 0, \ldots, 34$.
3. $\sigma_{\mathrm{avg}}$: First, calculate the standard deviation $\sigma^t$ for $R_{\mathrm{avg}}^{i,t}$ for $i = 1, 2, 3$, then calculate the average of all $\sigma^t$ as $\sigma_{\mathrm{avg}}$.

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

TABLE II
NMPC BASED DATASET

| Task | Average Reward | Reward Standard Deviation | Average Steps |
|---|---|---|---|
| Diagonal Folding | 95.1 | 2.4 | 4.4 |
| Central-Axis Folding | 87.1 | 3.5 | 6.1 |
| Flattening | 80.9 | 6.7 | 9.2 |

4. $Rank_{\text{avg}}$: Ranking all algorithms based on $R_{\text{avg}}$.
5. $Rank_{\sigma}$: Ranking all algorithms based on $\sigma_{\text{avg}}$. Lower rankings demonstrate higher stability.

### B. Experiment Settings for Verifying the Effectiveness of the NMPC Demonstration Dataset

We built a $6 \times 6$ point-mass spring-damper model, collected demonstration data, conducted 500 NMPC simulation runs per task and selected the top 100 rounds with the highest final rewards for each task to form $\mathcal{D}_{\text{NMPC}}$. Table II analyzes $\mathcal{D}_{\text{NMPC}}$ for three tasks. Compared to Table I, $\mathcal{D}_{\text{NMPC}}$ has similar average rewards and standard deviations but more average steps to complete tasks, showing NMPC strategies can finish tasks with lower efficiency. In this experiment, we use the same settings, metrics, and numbering as in Section IV-B. The $\mathcal{D}_{\text{NMPC}}$ generated by the NMPC algorithm serves as the demonstration training set for the FADERL model. Three statistical metrics (Reward Cosine Similarity (RCS), Reward Dynamic Time Warping (RDT), and Reward Pearson Correlation (RPC) of the average reward sequence $R^t_{\text{avg}}$ for a single test cycle) are used to compare the effectiveness of FADERL models trained with $\mathcal{D}_{\text{NMPC}}$ and $\mathcal{D}_{\text{demo}}$. We calculate these metrics for the average reward and standard deviation in a single test cycle. There are also corresponding metrics for the standard deviation: Standard Deviation Cosine Similarity (SCS), Standard Deviation Dynamic Time Warping (SDT), and Standard Deviation Pearson Correlation (SPC) of the standard deviation of reward sequences obtained with different random seeds.

### C. Physical Experiment for Deformable Object Robot Manipulation

Hand-eye calibration is used to determine the spatial relationship between the camera coordinate system and the robot coordinate system. Endpoint detection is utilized to extract key endpoints from the fabric's image. Optical flow tracking is employed for real-time tracking of the fabric endpoints' positions. The specific processes are as follows:

*1) Hand-Eye Calibration:* Easy-eye-hand software package.

*2) Endpoint Detection:* In the initial task stages, endpoint detection algorithms extract pixel coordinates. For non-initial folding states, the previous placement point $\boldsymbol{p}_{\text{p}_{t-1}}$ helps determine the baseline grasp point. Optical flow algorithms track other endpoints. After coordinate transformation and calibration, 3D positions in the robot coordinate system are obtained. An endpoint recognition algorithm, based on Canny edge detection and Douglas-Peucker polygon approximation, extracts key endpoints. It converts RGB to grayscale, applies Gaussian blur and Canny edge detection, refines edges, and simplifies the contour. Finally, we obtain a simplified contour containing a series of points, denoted as $\boldsymbol{C} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n)$. To select the furthest $k$ endpoints from the simplified contour $\boldsymbol{C}$ (where $k = 4$ for folding tasks and $k = 8$ for flattening tasks), this study devises Maximum Minimum Distance Vertices Selection method (MMDVS). For any $k$ points $\boldsymbol{p}_{r_1}, \boldsymbol{p}_{r_2}, \cdots, \boldsymbol{p}_{r_k}$ on the contour, a set $\boldsymbol{S}$ is contained all possible pairs of points. Then, the Euclidean distance between each pair is calculated and the minimum distance $d_{min}$ is gotten. By traversing all possible combinations of $k$ points in $\boldsymbol{C}$, the group of points with the maximum minimum distance $d_{min}$ are the desired $k$ endpoints $\boldsymbol{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_k)$. If the number of endpoints is less than $k$, we adjust the relevant parameters of the Douglas-Peucker algorithm to include new endpoints until a sufficient number of endpoints are obtained. The fabric area is also calculated using OpenCV functions.

*3) Optical Flow Tracking:* A designed optical-flow-based algorithm tracks the fabric's shape. Endpoints are classified into non-baseline and baseline grasp points. Non-baseline points are tracked based on previous-frame positions if tracking fails, and baseline points use the previous placement point as a new anchor after placement. The comprehensive experimental procedure is shown in Fig. S4. At the start, the robot and fabric are in the initial state. Then, the system loops: captures and analyzes the fabric state to form a comprehensive vector, inputs it into the pre-trained FADERL model to generate instructions, the robot executes actions, and after operation, the system collects and updates the state vector for the next step. The physical experimental setup consists of a workspace with dimensions of 100 cm $\times$ 80 cm. The camera (Intel RealSense D435i) is positioned 70 cm above the working surface, providing a clear view of the entire workspace. This configuration ensures sufficient coverage for fabric manipulation tasks while maintaining adequate resolution for endpoint detection. The robot base (UR5e) is positioned at a distance of 40 cm from the edge of the workspace, allowing sufficient reach for all fabric manipulation tasks.

## V. EXPERIMENT RESULTS

### A. Simulation Results of the Improved FADERL Algorithm With Human Demonstrations

Due to the page limitation, detailed results are presented in the supplementary material. Table S1 lists experiment numbers. Table S2 shows 8 control-group designs, evaluating the effectiveness of HTSK, GABC, and CPL in FADERL by comparing algorithms' performance. It also lists learning styles and expected results in Fig. 7 and Fig. S5. Fig. 7 and Fig. S5 show the dynamic changes of $R^t_{\text{avg}}$ for various algorithms under three tasks. Reward curves are smoothed. Tables S3-S5 list $R_{\text{avg}}$ and $Rank_{\text{avg}}$, and Tables S6-S8 list $\sigma_{\text{avg}}$ and $Rank_{\sigma}$. The last columns show metric averages. Table III gives global average metrics, highlighting optimal indicators in bold, with FADERL underlined.

From these figures and tables, we can derive key conclusions: Algorithm 1 (FADERL) excels in all simulations. HTSK boosts algorithm performance; red-marked (using HTSK)

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

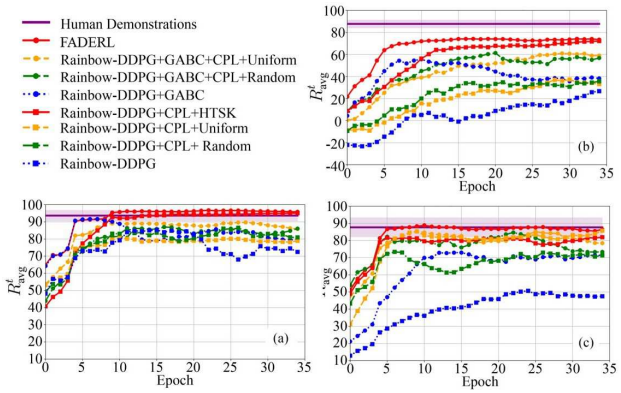



Fig. 7. Reward curve. (a) Folding along the diagonal. (b) Folding along the central-axis. (c) Flattening.

TABLE III

GLOBAL PERFORMANCE METRICS AND RANKINGS OF ALGORITHMS

| Algorithm Setting | Global Average Reward | Global Average Standard Deviation | Average Reward Rank | Average Standard Deviation Rank |
|---|---|---|---|---|
| 1 | **76.3** | **4.8** | **1.1** | **7.1** |
| 2 | 37.9 | 10.6 | 7.5 | 4.8 |
| 3 | 57.8 | 8.1 | 4.7 | 5.2 |
| 4 | 67.4 | 6.7 | 2.7 | 6.4 |
| 5 | 52.2 | 14.0 | 5.7 | 2.7 |
| 6 | 52.4 | 12.7 | 5.9 | 2.7 |
| 7 | 60.1 | 11.2 | 3.6 | 3.9 |
| 8 | 57.1 | 11.9 | 4.8 | 3.3 |

curves usually have advantages, outperforming random-selection algorithms, especially under stricter operation limits. GABC has a significant positive impact; circle-marked (using GABC) curves generally outperform square-marked ones, and GABC-integrated algorithms surpass non-GABC ones in most experiments. CPL's effectiveness is verified; its performance depends on the grasping-point selection strategy and operation constraints. Looser constraints and more stable strategies enhance CPL's effect. Finally, FADERL has the best performance across all metrics, achieving a 2.01-fold increase in global average reward and reducing the global average standard deviation to 45% of the baseline algorithm (Rainbow-DDPG).

### *B. Results of the Experiment for Verifying the Effectiveness of the NMPC Demonstration Dataset*

Fig. 8 and Fig. S6 show the curves of $R^t_{\text{avg}}$ for FADERL models trained with $\mathcal{D}_{\text{NMPC}}$ and $\mathcal{D}_{\text{demo}}$ for diagonal folding, central-axis folding, and flattening tasks. Tables S9-S11 display the performance of $\mathcal{D}_{\text{NMPC}}$-assisted FADERL models in $R_{\text{avg}}$, $\sigma_{\text{avg}}$, and the performance ratio compared to $\mathcal{D}_{\text{demo}}$-assisted models. In diagonal folding, FADERL learns effectively with either dataset. In central-axis folding and flattening, as difficulty increases, performance differences between the two datasets'-assisted FADERL models become significant. In simple tasks, $\mathcal{D}_{\text{NMPC}}$-assisted FADERL learns fast and may outperform $\mathcal{D}_{\text{demo}}$-assisted models, likely because NMPC performs well with sufficient steps. In challenging tasks, $\mathcal{D}_{\text{NMPC}}$-assisted FADERL generally underperforms, possibly due to NMPC's difficulty in completing tasks within limited steps. This aligns with NMPC's higher average steps in Table II. Results of Experiments 3.1 and 3.6 show stochastic factors.

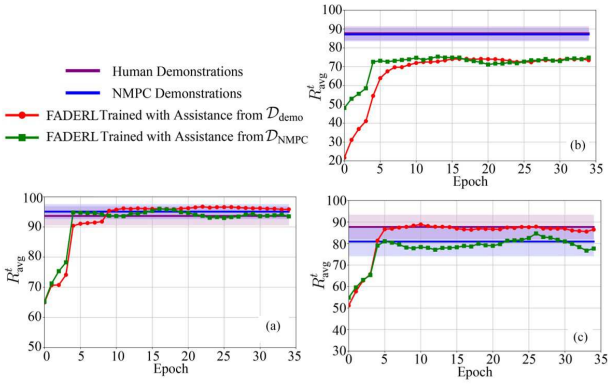



Fig. 8. Performance comparison of FADERL trained with assistance from $\mathcal{D}_{\text{NMPC}}$ and $\mathcal{D}_{\text{demo}}$. (a) Folding along the diagonal. (b) Folding along the central-axis. (c) Flattening.

TABLE IV

COMPARISON OF GLOBAL PERFORMANCE METRICS FOR FADERL TRAINED WITH $\mathcal{D}_{\text{NMPC}}$ AND $\mathcal{D}_{\text{DEMO}}$

| Demonstration Dataset | Global Average Reward | Global Average Standard Deviation |
|---|---|---|
| $\mathcal{D}_{\text{demo}}$ | 76.3 | 4.8 |
| $\mathcal{D}_{\text{NMPC}}$ | 76.1 | 4.0 |

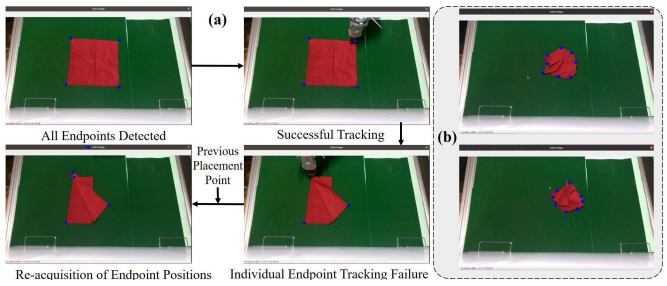



Fig. 9. (a) Optical flow tracking results. (b) Endpoint detection results under crumpled state.

Table IV compares overall performance metrics. The $\mathcal{D}_{\text{NMPC}}$-assisted FADERL model achieves 99.7% of the global average reward and 83.3% of the global average standard deviation of the $\mathcal{D}_{\text{demo}}$-assisted model, indicating similar performance levels. From Table S12, $\mathcal{D}_{\text{NMPC}}$ and $\mathcal{D}_{\text{demo}}$-trained FADERL models show significant similarity in the $R^t_{\text{avg}}$ sequence (notably in RCS and RPC), highlighting consistency. However, there are significant differences in the standard deviation of reward sequences under different random seeds, which may be due to experimental randomness and stylistic differences between NMPC and human-operated strategies.

### *C. Results of Physical Experiments on Visual Processing and Robot Manipulation*

The endpoint recognition algorithm targets two fabric states: fully flattened and fully wrinkled (Fig. 9). It can accurately identify 4 endpoints in the flattened state and 8 representative endpoints in the wrinkled state, providing accurate initial positions for optical flow tracking. The optical flow tracking algorithm tracks fabric endpoints in folding tasks. It can

IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.

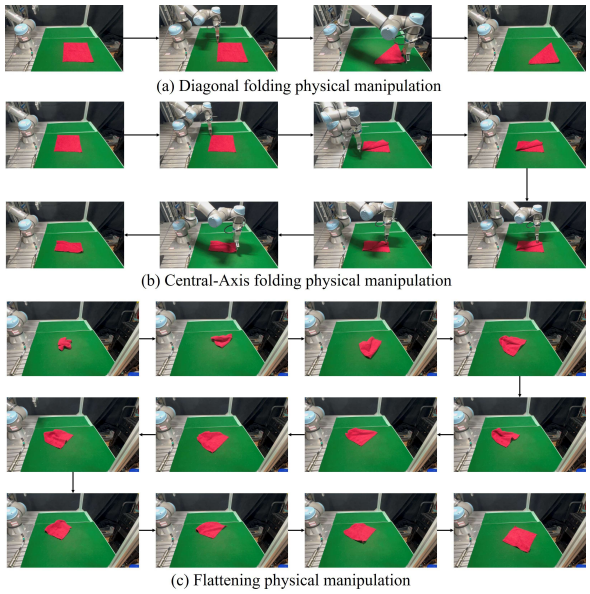



Fig. 10. Physical manipulation procedure.

TABLE V
PHYSICAL EXPERIMENT RESULTS

| Task | Completion $\geq 0.6$ | Completion $\geq 0.8$ | Success Rate | Average Steps |
|---|---|---|---|---|
| Diagonal Folding | 93.3% | 90.0% | 83.3% | 1.1 |
| Central-Axis Folding | 90.0% | 86.7% | 80.0% | 3.9 |
| Flattening | 100.0% | 100.0% | 96.7% | 13.5 |

accurately track as the fabric shape changes, but occlusion by the end-effector can cause tracking failures. After operation, the placement point is used as the new reference grasping point. The experimental operation process is illustrated in Fig. 10. In physical experiments, we use indicators instead of directly measuring endpoint distances:

1. Task Completion Rate: Set target fabric shapes for different tasks and calculate similarity with the actual shape using OpenCV's "cv2.matchShapes()" as the task completion rate.

2. Success Rate: An experiment is successful when the final task completion rate exceeds 0.9.

3. Average Steps: The average number of actions required for the robot to complete a specific task measured the efficiency of the robot's operations.

As shown in Table V, for diagonal folding, 93.3% of trials have a score over 0.6, 90.0% over 0.8, with an 83.3% success rate and 1.1 average steps. For central-axis folding, 90.0% reach over 0.6, 86.7% over 0.8, with an 80.0% success rate and 3.9 average steps. For flattening, all trials meet over 0.6 and 0.8 standards, with a 96.7% success rate and 13.5 average steps. Despite the most steps, its high success rate is due to error tolerance. Overall, all tasks have high success rates, indicating good experimental performance. Our system operates at real-time performance with FADERL policy inference taking approximately 0.1 ms per step, as evidenced by the smooth and continuous motion observed in our supplementary video. This is significantly faster than current large VLA models, which typically require tens to hundreds of milliseconds per inference step.

### D. Discussion

In the context of robotic manipulation tasks for deformable objects, this article addresses the inefficiency of traditional RL methods by proposing the FADERL algorithm. To avoid high costs associated with traditional human teaching methods, a low-cost data augmentation method based on NMPC is introduced. The effectiveness of the proposed methods is validated through three experimental scenarios, both in simulation and real-world experiments. Extensive ablation studies are conducted to substantiate the rationality and efficacy of the algorithms.

Compared to similar research, Matas et al. [22] required nearly 80,000 interactions between the robot and the environment to complete the learning process; Jangir et al. [23] needed approximately 260,000 rounds of interaction data to train their agent; Yang et al. [26] utilized 28,000 pairs of images and actions collected via teleoperation to train a DNN as an end-to-end policy for folding a single towel. This study simplifies the data acquisition process and achieves comparable or even higher success rates than the aforementioned studies, providing novel insights and contributions for future tasks of a similar nature. Moreover, compared with the currently popular approaches based on large models for robot manipulation, the algorithm proposed in this paper has the advantages of being lightweight, requiring low computational resources, and being able to provide task-specific customization and efficient adaptability when handling specific tasks.

## VI. CONCLUSION

This article studies deformable object robot manipulation based on demonstration-enhanced RL. To boost RL learning efficiency, it improves algorithm utilization of demonstration data, proposing the FADERL algorithm and collecting $\mathcal{D}_{\text{demo}}$. It first trains the HTSK fuzzy system with demonstration data for grasping-point selection, then proposes GABC to enhance demonstration-data utilization, and finally uses CPL to integrate HTSK and GABC-improved Rainbow-DDPG, forming FADERL. Simulation experiments verify its effectiveness. Compared to Rainbow-DDPG, FADERL achieves a 2.01-fold increase in global average reward and reduces the global average standard deviation to 45%. A low-cost NMPC-based data augmentation method is proposed. Based on a spring-mass model, NMPC controls tasks in simulation, using high-reward trajectories as data. Simulation shows that the FADERL model trained with $\mathcal{D}_{\text{NMPC}}$ achieves 99.7% of the global average reward and 83.3% of the global average standard deviation of the model trained with $\mathcal{D}_{\text{demo}}$, indicating $\mathcal{D}_{\text{NMPC}}$'s comparable effectiveness. In physical experiments, success rates of 83.3%, 80%, and 96.7% in three tasks validate the method.

While our approach demonstrates promising results, it is important to acknowledge that our current experimental validation primarily focuses on standard red towels under controlled lighting conditions. This choice reflects our targeted application scenarios in household and service robotics where consistent material properties simplify the initial validation of our core algorithmic contributions. Notably, the modular

**IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.**

design of our visual processing pipeline allows for straightforward adaptation to different fabric colors and shapes by simply modifying the edge detection component. For instance, when handling dark-colored fabrics, we can replace the Canny edge detector with more robust alternatives such as the Structured Forests edge detector, while maintaining the same core control framework. Similarly, for non-rectangular geometries, the Maximum Minimum Distance Vertices Selection (MMDVS) method can be extended to identify key points on arbitrary shapes without requiring fundamental changes to the NMPC or FADERL algorithms. This modularity represents a key strength of our lightweight approach compared to monolithic large-model solutions that would require complete retraining for such variations. Future work will build upon this modular framework to systematically explore the adaptability of our approach across diverse material properties. Specifically, we plan to investigate adaptive perception techniques that can automatically select appropriate edge detection strategies based on real-time analysis of fabric color and texture while maintaining the computational efficiency.

## REFERENCES


[1] J. Zhu et al., "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robot. Autom. Mag.*, vol. 29, no. 3, pp. 67–77, Sep. 2022.

[2] R. Shukla et al., "Performing efficient and safe deformable package transport operations using suction cups," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Abu Dhabi, United Arab Emirates, Oct. 2024, pp. 11835–11842.

[3] T. Fu, C. Li, J. Liu, F. Li, C. Wang, and R. Song, "FlingFlow: LLM-driven dynamic strategies for efficient cloth flattening," *IEEE Robot. Autom. Lett.*, vol. 9, no. 10, pp. 8714–8721, Oct. 2024.

[4] D. Kang, S. Hong, S.-J. Kim, H. Choi, K. Kim, and J. Jang, "Robotics-assisted modular assembly of bioactive soft materials for enhanced organ fabrication," *Virtual Phys. Prototyping*, vol. 19, no. 1, Aug. 2024, Art. no. e2390484.

[5] J. Chen, G. Ning, L. Ma, and H. Liao, "Autonomous deformable tissue retraction system based on 2-D visual representation and asymmetric reinforcement learning for robotic surgery," *IEEE Trans. Med. Robot. Bionics*, vol. 7, no. 2, pp. 595–606, May 2025.

[6] H. Luo and Y. Demiris, "TSL: Tracking deformable linear objects for bimanual shoe lacing," *IEEE Robot. Autom. Lett.*, vol. 10, no. 8, pp. 8212–8219, Aug. 2025.

[7] F. Zhang and Y. Demiris, "Learning garment manipulation policies toward robot-assisted dressing," *Sci. Robot.*, vol. 7, no. 65, Apr. 2022, Art. no. eabm6010.

[8] J. Ye, W. Li, S. Khan, and G. S. Chirikjian, "RaggeDi: Diffusion-based state estimation of disordered rags, sheets, towels and blankets," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Atlanta, GA, USA, May 2025, pp. 10371–10377.

[9] J. Huang, T. Fukuda, and T. Matsuno, "Model-based intelligent fault detection and diagnosis for mating electric connectors in robotic wiring harness assembly systems," *IEEE/ASME Trans. Mechatronics*, vol. 13, no. 1, pp. 86–94, Feb. 2008.

[10] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 479–484.

[11] N. A. Strohmeyer, J. H. Park, B. P. Murphy, and F. Alambeigi, "A semi-autonomous data-driven shared control framework for robotic manipulation and cutting of an unknown deformable tissue," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Japan, May 2024, pp. 9881–9886.

[12] A. Jevtic et al., "Personalized robot assistant for support in dressing," *IEEE Trans. Cognit. Develop. Syst.*, vol. 11, no. 3, pp. 363–374, Sep. 2019.

[13] J. Zhu, M. Gienger, G. Franzese, and J. Kober, "Do you need a hand?—A bimanual robotic dressing assistance scheme," *IEEE Trans. Robot.*, vol. 40, pp. 1906–1919, 2023.

[14] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey," *Int. J. Robot. Res.*, vol. 37, no. 7, pp. 688–716, Jun. 2018.

[15] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Sci. Robot.*, vol. 6, no. 54, May 2021, Art. no. eabd8803.

[16] J. Fu, I. Burzo, E. Iovene, J. Zhao, G. Ferrigno, and E. De Momi, "Optimization-based variable impedance control of robotic manipulator for medical contact tasks," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–8, 2024.

[17] D. Navarro-Alarcon and Y.-H. Liu, "Fourier-based shape servoing: A new feedback method to actively deform soft objects into desired 2-D image contours," *IEEE Trans. Robot.*, vol. 34, no. 1, pp. 272–279, Feb. 2018.

[18] Z. Li, C. Zeng, Z. Deng, Q. Xu, B. He, and J. Zhang, "Learning variable impedance control for robotic massage with deep reinforcement learning: A novel learning framework," *IEEE Syst., Man, Cybern. Mag.*, vol. 10, no. 1, pp. 17–27, Jan. 2024.

[19] W. Wang, C. Zeng, H. Zhan, and C. Yang, "A novel robust imitation learning framework for complex skills with limited demonstrations," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 3947–3959, 2025.

[20] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, and K. Goldberg, "SpeedFolding: Learning efficient bimanual folding of garments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Japan, Oct. 2022, pp. 1–8.

[21] A. Colomé and C. Torras, "Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 602–615, Jun. 2018.

[22] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Proc. 2nd Conf. Robot Learn.*, Zürich, Switzerland, 2018, pp. 734–743.

[23] R. Jangir, G. Alenyá, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Paris, France, May 2020, pp. 4630–4636.

[24] C. Zeng, S. Li, Z. Chen, C. Yang, F. Sun, and J. Zhang, "Multifingered robot hand compliant manipulation based on vision-based demonstration and adaptive force control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 5452–5463, Sep. 2023.

[25] E. Pignat and S. Calinon, "Learning adaptive dressing assistance from human demonstration," *Robot. Auto. Syst.*, vol. 93, pp. 61–75, Jul. 2017.

[26] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable folding task by humanoid robot worker using deep learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 397–403, Apr. 2017.

[27] A. Cherubini, J. Leitner, V. Ortenzi, and P. Corke, "Towards vision-based manipulation of plastic materials," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Spain, Oct. 2018, pp. 485–490.

[28] J. Zhu, M. Gienger, and J. Kober, "Learning task-parameterized skills from few demonstrations," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4063–4070, Apr. 2022.

[29] D. Vodolazskii, M. Li, H. Wu, and H. Handroos, "A review of robotic manipulation of deformable objects with imitation learning techniques: Progress and outlook," in *Proc. 11th Int. Conf. Electr. Eng., Control Robot. (EECR)*, China, Apr. 2025, pp. 9–20.

[30] M. Jin Kim et al., "OpenVLA: An open-source vision-language-action model," 2024, *arXiv:2406.09246*.

[31] A. Brohan et al., "RT-2: Vision-language-action models transfer Web knowledge to robotic control," in *Proc. Conf. Robot Learn.*, Atlanta, GA, USA, 2023, pp. 2165–2183.

[32] J. Wen et al., "TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation," 2024, *arXiv:2409.12514*.

[33] G. R. Team et al., "Gemini robotics: Bringing AI into the physical world," 2025, *arXiv:2503.20020*.

[34] C. Cheang et al., "GR-3 technical report," 2025, *arXiv:2507.15493*.

[35] H. Wang et al., "Grasping state analysis of soft manipulator based on flexible tactile sensor and high-dimensional fuzzy system," *IEEE/ASME Trans. Mechatronics*, vol. 30, no. 3, pp. 2165–2176, Jun. 2025.

[36] Y. Cui, Y. Xu, R. Peng, and D. Wu, "Layer normalization for TSK fuzzy system optimization in regression problems," *IEEE Trans. Fuzzy Syst.*, vol. 31, no. 1, pp. 254–264, Jan. 2023.

[37] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to manipulate deformable objects without demonstrations," in *Proc. 16th Robot., Sci. Syst.*, 2020, pp. 1–11. [Online]. Available: https://roboticsproceedings.org/rss16/p065.html

**IEEE Transactions on Automation Science and Engineering (T-ASE) paper, presented at ICRA 2026, Vienna, Austria.**

[38] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actorcritic methods," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, 2018, pp. 1587–1596.

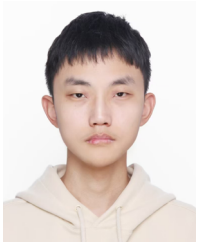
[39] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

**Haoyuan Wang** (Graduate Student Member, IEEE) received the B.E. degree in automation from the University of Electronic Science and Technology of China, Chengdu, China, in 2020. He is currently pursuing the Ph.D. degree in intelligence science and technology with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China. His research interests include machine learning, grasping and manipulation, and robot tactile perception.

**Zihao Dong** received the B.S. degree in automation from Huazhong University of Science and Technology, Wuhan, China, in 2021, and the master's degree in artificial intelligence major from the Huazhong University of Science and Technology, Wuhan, China, in 2024. He is currently an Assistant Engineer with China Academy of Aerospace System and Innovation. His current research interests include robotic arm modeling and intelligent control, reinforcement learning techniques, and computer vision.

**Tong Zhu** received the B.E. degree in artificial intelligence from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2025. She is currently pursuing the M.S. degree in artificial intelligence with the School of Artificial Intelligence and Automation, HUST. Her research interests include reinforcement learning and robotic sorting operations.

**Hongliang Lei** received the B.S. degree in automation from Huazhong University of Science and Technology from Huazhong University of Science and Technology in 2024, Wuhan, China, where he is currently pursuing the Ph.D. degree in intelligence science and technology. His current research interests include robotic arm operation and manipulation.



**Weizhuang Shi** received the B.S. degree in automation from Huazhong University of Science and Technology in 2024. He is currently pursuing the master's degree in control engineering with Huazhong University of Science and Technology. His current research interests include robotic arm operation and imitation learning.

**Zejia Zhang** received the B.S. degree in automation from Huazhong University of Science and Technology in 2023. She is currently pursuing the master's degree in artificial intelligence with Huazhong University of Science and Technology. Her current research interests include intent recognition, eye-controlled robotic arm, and robotic arm operation.

**Wei Luo** received the B.Sc. degree in computer science from Wuhan University of Technology, Wuhan, China, in 2002, and the M.Sc. and Ph.D. degrees in computer science from Huazhong University of Science and Technology, Wuhan, in 2005 and 2008, respectively. He is currently an Associate Researcher with the Department of Innovation Center, China Ship Development and Design Center, Wuhan. His research interests include unmanned systems, robotics, real-time/embedded systems, and artificial intelligence.

**Weiwei Wan** (Senior Member, IEEE) is currently an Associate Professor with the School of Engineering Science, The University of Osaka, Japan. His research interests include planning and learning for smart manufacturing and robotic manipulation.

**Xinxing Chen** (Member, IEEE) received the B.E. degree in optical and electronic information engineering and the Ph.D. degree in control science and engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2015 and 2020, respectively. She was with Southern University of Science and Technology, Shenzhen, China, as a Post-Doctoral Fellow and an Associate Researcher from 2020 to 2024. She is currently an Assistant Professor with the School of Artificial Intelligence and Automation, HUST. Her main research interests include mobile robotics and walking assistive robots.

**Jian Huang** (Senior Member, IEEE) received the bachelor's and M.E. degrees in control theory and control engineering and the Ph.D. degree in control science and engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1997, 2000, and 2005, respectively. From 2006 to 2008, he was a Post-Doctoral Researcher with the Department of Micro–Nano System Engineering and the Department of Mechano-Informatics and Systems, Nagoya University, Nagoya, Japan. He is currently a Full Professor with the School of Artificial Intelligence and Automation, HUST. His main research interests include rehabilitation robots, robotic assembly, networked control systems, and bioinformatics.