Global Model Learning for Large Deformation Control of Elastic Deformable Linear Objects: An Efficient and Adaptive Approach

Mingrui Yu, Student Member, IEEE, Kangchen Ly, Hanzhong Zhong, Shiji Song, Senior Member, IEEE, and Xiang Li, Member, IEEE

Abstract—The robotic manipulation of deformable linear objects (DLOs) has broad application prospects in many fields. However, a key issue is to obtain the exact deformation models (i.e., how robot motion affects DLO deformation), which are hard to theoretically calculate and vary among different DLOs. Thus, the shape control of DLOs is challenging, especially for large deformation control that requires global and more accurate models. In this article, we propose a coupled offline and online data-driven method for efficiently learning a global deformation model, allowing for both accurate modeling through offline learning and further updates for new DLOs via online adaptation. Specifically, the model approximated by a neural network is first trained offline on random data, then seamlessly migrated to the online phase, and further updated online during actual manipulation. Several strategies are introduced to improve the model’s efficiency and generalization ability. We propose a convex-optimization-based controller and analyze the system’s stability using the Lyapunov method. Detailed simulations and real-world experiments demonstrate that our method can efficiently and precisely estimate the deformation model and achieve the large deformation control of untrained DLOs in 2-D and 3-D dual-arm manipulation tasks better than the existing methods. It accomplishes all 24 tasks with different desired shapes on different DLOs in the real world, using only simulation data for the offline learning.

Index Terms—Deformable linear objects (DLOs), model learning, robotic manipulation, shape control.

I. INTRODUCTION

DEFORMABLE linear objects (DLOs) refer to deformable objects in one dimension, such as ropes, elastic rods, wires, cables, etc. The demand for manipulating DLOs is reflected in many applications, and a significant amount of research efforts have been devoted to the robotic solutions to these applications [1], [2], [3], [4]. For example, wires are manipulated to assemble devices in 3C manufacturing [5]; belts are manipulated in assemblies of belt drive units [6]; and in surgery, sutures are manipulated to hold tissue together [7].

The manipulation tasks of DLOs can be divided into two categories [8]. In the first category, the goals are not about the exact shapes of DLOs; rather, they concern high-level conditions, such as tangling or untangling knots [9], [10], obstacle avoidance [11], [12], following and insertion [13], etc. The second category concerns manipulating DLOs to desired shapes, where a key challenge is to estimate the unknown deformation models (i.e., how the robot motion affects the DLO shapes) with sufficient accuracy [14]. This article focuses on the shape control tasks.

It is very challenging to obtain the exact deformation models of DLOs. First, they are hard to calculate theoretically. Some analytical modeling methods can be used to model DLOs, such as mass–spring systems, position-based dynamics, and finite-element methods [2], [15]. However, all are approximate models and require the accurate parameters of DLOs, which are difficult to acquire in the real world. As a result, data-driven approaches have been widely applied to learn the deformation...
models without studying the complex physical dynamics. Second, the deformation model is nonlinear with respect to the DLO configuration, making it usually data inefficient to learn an accurate global model effective for any DLO shape. Note that such a global model is essential for the large deformation control of DLOs, where the initial and final shapes can be very different. Third, the deformation models may vary significantly among different DLOs owing to different lengths, thicknesses, and materials. It is impractical to pay a huge time cost to collect new data of the new DLO before every manipulation task. Thus, the adaptiveness to different DLOs must be considered.

The existing data-driven methods to obtain the deformation model can mainly be divided into purely offline and purely online methods. For the purely offline methods, the most common one is to first learn a forward kinematics model (FKM) offline and then use the model-predictive control (MPC) in manipulations [16], [17], [18], [19], [20]. Reinforcement learning (RL) methods have also been studied [21], [22]. Although they can achieve the large deformation control of a well-trained DLO, they usually require a large amount of data for model learning and may have trouble manipulating a different untrained DLO. Apart from these offline approaches, some studies have used purely online methods to estimate a local linear deformation model (Jacobian matrix) of the manipulated DLO [23], [24], [25], [26], [27], [28]. Since they are executed purely online, they can be applied to any new DLO. However, because only a small amount of local data can be utilized, these estimated models are less accurate and only effective in local configurations, making them only able to handle tasks with local and small deformation. While both offline learning and online learning have advantages, finding a solution to utilize them to complement each other effectively is not trivial.

In this article, we consider the problem of the large deformation control of elastic DLOs, where the initial configurations of DLOs are far from the desired shapes. The “large” here is relative to the existing works. To achieve it, we propose a coupled offline learning and online adaptation method for efficiently learning the global deformation model, as illustrated in Fig. 1. This complementary scheme allows more accurate modeling through offline learning and further updating for new DLOs via online adaptation. Specifically, we use a radial basis function neural network (RBFN) to globally model the nonlinear mapping from the current state to the current local linear deformation model (a locally effective Jacobian matrix). The RBFN is first offline trained on collected random data, then migrated to the online phase as an initial estimation, and finally further updated to adapt to the manipulated DLO during actual manipulation. Hence, both the advantages of offline and online learning are well explored and seamlessly incorporated. In addition, we introduce several strategies to improve the model’s generalization ability and training efficiency, such as scale normalization and rotation data augmentation. We also propose a convex-optimization-based feedback control law, which considers the singularity of the Jacobian and constrains the robots not to overstretch the DLO. The stability of the closed-loop system and the convergence of task errors are analyzed using the Lyapunov method. Exhaustive simulation and real-world experiments are carried out to validate the proposed method. The video and code are available at the project website.\(^1\)

Our key contributions are highlighted as follows:
1) We prove that the deformation model of DLOs can be globally described by a nonlinear mapping from the DLO configuration to a local Jacobian matrix in quasi-static manipulations. Such models can be learned more data efficiently than theFKMs.
2) We propose a coupled offline and online method to efficiently learn the global deformation model, which first achieves both stable large deformation control and effective adaptation to new DLOs.
3) We conduct detailed simulation and real-world experiments to demonstrate the outperformance of the proposed method over the existing works.

This work is an extension of our previous work presented in [29], which proposed a preliminary coupled offline and online model learning method for DLO shape control. The improvements include: 1) proposing new model modifications and training strategies to improve the model’s generalization ability and data efficiency; 2) proposing a new controller based on convex optimization, which considers the singularity of the Jacobian and constrains the robots not to overstretch the DLO; 3) proposing a new robust online model updating law with detailed rigorous stability analysis; and 4) conducting more detailed simulation studies and real-world 2-D and 3-D dual-arm manipulation experiments.

II. RELATED WORK

A. DLO Shape Control Tasks

The shape control tasks of DLOs can be further divided into two types. The first type concerns manipulating soft ropes placed on tables, where the ropes are so soft that their deformation shapes can be held by friction of tables without being grasped by robots (i.e., such deformation can be considered as plastic deformation). Therefore, the robots can move the DLO to the desired shape by executing a series of pick-and-place actions at any point on the DLO [16], [17], [18], [19], [20], [21]. The second type is about manipulating stiffer DLOs such as flexible rods and cables, in which their deformation under forces from robots is mainly elastic deformation [22], [23], [24], [25], [26], [27], [28]. The robots grasp only the ends of DLOs to control the internal shapes, so the deformation model is essential. Moreover, the robot degrees of freedom (DoFs) can be up to 12 in 3-D dual-arm manipulations, making modeling and control more challenging. This article focuses on the second type of shape control tasks: manipulating elastic DLOs. While the existing works usually consider local and small deformation, this article deals with much larger deformation.

B. Existing Approaches

1) With Analytical Models: The analytical modeling of DLOs has been researched over the past several decades [2].

\(^1\) [Online]. Available: https://mingrui-yu.github.io/shape_control_DLO_2
Some works on shape control were based on analytical models. In [30] and [31], the static equilibrium configurations of elastic rods were analyzed using a geometric model, and the simulated manipulation was based on planning a proper path through the set of equilibrium shapes. In [32], an energy-based elastic rod model was utilized for the dynamic simulation of DLOs, and a heuristic model-free controller was proposed for DLO shaping. The finite-element model (FEM) simulation of DLOs was used for open-loop shape control in [33], and a reduced FEM was used for closed-loop shape control in [34]. The application of these methods is limited because the models usually require a large amount of computation and accurate DLO parameters (such as the cross-section area, Young modulus, shear modulus, etc.), which are hard and cumbersome to obtain in the real world.

2) With Demonstrations or RL: Recently, data-driven approaches have been applied to the shape control of DLOs, dispensing with analytical modeling. In [35], [36], and [37], the shaping of DLOs was addressed by learning from human demonstrations. Robots could reproduce human actions for specific tasks. RL has also been applied to learn control policies in an end-to-end manner. A simulated benchmark of RL algorithms for deformable object manipulation was presented in [21], in which rope straightening and shaping tasks were studied. RL policies for the shape control of elastoplastic DLOs in simulation were learned in [22], and a simulation sandbox for DLO manipulation was introduced in [8]. RL-based methods for DLO manipulation are in the early stages of research. Like other RL applications, these methods suffer from considerably high training expenses and challenging transfer from simulation to real-world scenarios. As a result, up to now, RL-based methods have been primarily studied in simulation only and are difficult to apply in the real world.

3) With Offline-Learned Forward Kinematics Models: Different from the end-to-end methods, many works first learn neural-network (NN)-based FKM s of DLOs offline and then use the MPC to control the shape. An FKM predicts the shape at the next time step based on the current shape and input action. In [20], an FKM in the image space was learned, and random-sampling-based planning was applied for control. In [16] and [17], an encoder from the image space to the latent space and an FKM in the latent space were jointly trained. A more robust and data-efficient approach is to estimate the DLO state first and then learn the FKM in the physical state space [18], [19]. Owing to the nonlinear DLO kinematics and complex NN architectures, the training of these models often requires tens of hours of data. Moreover, their generalization to different untrained DLOs cannot be guaranteed.

4) With Online-Estimated Local Models: To control the shape of unknown objects, a series of methods tackle the shape control problem based on purely online estimation of local linear deformation models of DLOs, in which a small change of the DLO is linearly related to a small movement of the manipulator by a locally effective estimated Jacobian matrix. The control input is directly calculated using the inverse of the Jacobian. The estimated Jacobian matrix was updated online using the Broyden update rule [23], the gradient descent method [24], or the (weighted) least-squares estimation on recent data in the current sliding window [25], [26], [27], [28]. Compared with the offline models, these online estimated models are only effective in local configurations and less accurate because only limited local data are utilized. Thus, they mostly handle tasks with local and small deformation. In addition, their estimated local models cannot be used for multistep predictions or even reused for new tasks.

5) With Offline + Online Model Learning: To leverage both the advantages of the offline and online learning, we proposed a preliminary coupled offline and online model learning method in our previous work [29]. Later, Wang et al. [38] proposed another scheme to combine the two phases. They first trained a nonlinear FKM in the offline phase and then used another local Jacobian model to compensate for the residual error of the FKM in the online phase. In contrast, in our method, by reformulating the deformation model as a global Jacobian model, the shortcomings of the FKM and the local Jacobian model are avoided, and the offline learning and online adaptation can be executed on the same model with seamless migration. Moreover, [38] dealt with local deformation in 2-D scenarios, while our method is validated on 3-D large deformation control tasks.

III. PRELIMINARIES

A. Problem Formulation

This article considers the quasi-static shape control of elastic DLOs. As illustrated in Fig. 2, the robots grasp the ends of the DLO and manipulate it to the desired shape. The overall shape of the DLO is represented by the positions of multiple features uniformly distributed along the DLO. The target points are chosen from the features, and the task is defined as moving the target points on the DLO to their corresponding desired positions by controlling the velocities of the end-effectors. The specific choice of the target points depends on the task needs.

Assumptions: The following assumptions are made for the DLO manipulation.

1) Only elastic deformation of the DLO will happen during manipulation.
2) The manipulation process is quasi-static, meaning that the shape of the DLO is determined by only its potential energy and no inertial effects during manipulation [24].
3) The stiffness matrix of the DLO is positive and full-rank around the equilibrium point [24].
4) The ends of the DLO have been rigidly grasped by the robot end-effectors. The velocities of the end-effectors can be kinematically controlled.

These assumptions are commonly used in the research of deformable object manipulation.

For the generalization and simplicity of writing, we formulate the problem as a 3-D dual-arm manipulation task in the following text. Note that our method can also be applied to other specific settings, such as 2-D or single-arm manipulations.

B. Notations

Some frequently used notations are described as follows:

1) $I$ is the identity matrix.
where the velocity vector of the DLO features can be locally linearly related to the velocity vector of the end-effectors using a Jacobian matrix. Note that the Jacobian matrix is varying during moving and needs to be constantly updated. Most of the previous works derive such models by assuming that the DLO shape can be determined by the configuration of the end-effectors using a function like \( \bar{\vec{x}} = h(\vec{r}) \) and differentiating it with respect to time [23], [28], [39]. However, in a global sense, the DLO shape cannot be uniquely determined by the configuration of the end-effectors [30]. In the next section, we derive the Jacobian-based model in another way, using the quasi-static assumption.

IV. DEFORMATION MODEL

A. Global Jacobian-Based Deformation Model

One key problem of DLO shape control is studying the mapping from the motion of the end-effectors to the motion of the DLO features, which is essential for model-based control. We also use the Jacobian to describe the local relationship, but in a global way. That is, we discover that the Jacobian matrix can be fully determined by the current state, so our model learns the mapping from the state \((\vec{x}, \vec{r})\) to the corresponding Jacobian matrix, which is specified as

\[
\dot{\vec{x}} = J(\vec{x}, \vec{r}) \nu.
\]

Note that here \( J(\cdot) \) is a globally effective function.

**Theorem 1**: Under the quasi-static assumption, the velocity vector of the features on the elastic DLO can be related to the velocity vector of the end-effectors as (3).

**Proof**: Denote the potential energy of the elastic DLO as \( E \), which is assumed to be fully determined by \( \vec{x} \) and \( \vec{r} \). In the quasi-static assumption, internal equilibrium holds at any state during the manipulation, where the DLO’s internal shape \( \vec{x} \) locally minimizes the potential energy \( E \) [30]. That is, \( \partial E / \partial \vec{x} = 0 \) at any state. Consider the DLO is moved from state \((\bar{\vec{x}}, \bar{\vec{r}})\) to state \((\bar{\vec{x}} + \delta \vec{x}, \bar{\vec{r}} + \delta \vec{r})\), where \( \delta \vec{x} \) and \( \delta \vec{r} \) are small movements of the features and grasped ends. Denote

\[
g(\vec{x}, \vec{r}) = \frac{\partial E}{\partial \vec{x}}, \quad \dot{\vec{A}}(\vec{x}, \vec{r}) = \frac{\partial^2 E}{\partial \vec{x} \partial \vec{r}}, \quad \vec{B}(\vec{x}, \vec{r}) = \frac{\partial^2 E}{\partial \vec{x} \partial \vec{r}}.
\]

Using the Taylor expansion and neglecting the higher order terms, we have

\[
g(\bar{\vec{x}} + \delta \vec{x}, \bar{\vec{r}} + \delta \vec{r}) \approx g(\bar{\vec{x}}, \bar{\vec{r}}) + \vec{A}(\bar{\vec{x}}, \bar{\vec{r}}) \delta \vec{x} + \vec{B}(\bar{\vec{x}}, \bar{\vec{r}}) \delta \vec{r}.
\]

With the assumption, \( g(\vec{x}, \vec{r}) = 0 \), so \( g(\bar{\vec{x}} + \delta \vec{x}, \bar{\vec{r}} + \delta \vec{r}) \approx g(\bar{\vec{x}}, \bar{\vec{r}}) = 0 \). In addition, \( \vec{A} \) and \( \vec{B} \) physically represent the stiffness matrices; assuming the DLO has a positive and full-rank stiffness matrix around the equilibrium point, matrix \( \vec{A} \) is invertible [24]. Then, it can be obtained that

\[
\delta \vec{x} \approx - (\vec{A}(\bar{\vec{x}}, \bar{\vec{r}}))^{-1} \vec{B}(\bar{\vec{x}}, \bar{\vec{r}}) \delta \vec{r}.
\]

Note that this equation holds for arbitrary state \((\bar{\vec{x}}, \bar{\vec{r}})\). In addition, in slow manipulations, \( \bar{\vec{x}} \approx \delta \vec{x} / \delta t \) and \( \bar{\vec{r}} \approx \delta \vec{r} / \delta t \) with small \( \delta t \). Thus, we have

\[
\dot{\vec{x}} \approx - (\vec{A}(\vec{x}, \vec{r}))^{-1} \vec{B}(\vec{x}, \vec{r}) \dot{\vec{r}}.
\]
where \( \dot{r} \) is the derivative of the configuration of the grasped ends with respect to time, which can be related to the velocity vector \( \nu \) by a matrix as

\[
\dot{r} = \begin{bmatrix} p_1 \\ q_1 \\ p_2 \\ q_2 \end{bmatrix} = \begin{bmatrix} I_{3 \times 3} & M(q_1) \\ I_{3 \times 3} & M(q_2) \end{bmatrix} \begin{bmatrix} v_1 \\ \omega_1 \\ v_2 \\ \omega_2 \end{bmatrix} C(\nu)
\]

(8)

where \( M \) is the matrix relating the derivative of the quaternion to the angular velocity vector, which is determined by the quaternion. Then, denoting \( - (A(x, r))^{-1} B(x, r) C(\nu) \) as \( J(x, r) \), we derive (3) and prove Theorem 1.

Note that (3) can be rewritten as

\[
\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_m \end{bmatrix} = \begin{bmatrix} J_1(x, r) \\ \vdots \\ J_m(x, r) \end{bmatrix} \nu \tag{9}
\]

where \( J_k(x, r) \) is the \( (3k - 1 + 1) \)th to \( (3k) \)th rows of \( J(x, r) \). Thus, it can be obtained that

\[
\dot{x}_k = J_k(x, r) \nu, \quad k = 1, \ldots, m
\]

(10)

where \( m \) is the number of the features. It indicates that different features correspond to different Jacobian functions. This formulation makes it convenient when choosing any subset of features as the target points in the manipulation tasks.

We emphasize that this Jacobian-based model is global because it is effective for any DLO state, which is essential for large deformation control. We then estimate it using a data-driven method based on an NN, where the input is the current state and the output is the Jacobian matrix.

### B. Model Modifications to Improve Generalization Ability

We make the following modifications to improve the model’s generalization ability on different DLOs in large deformation control tasks.

First, it can be noticed that the Jacobian is translation invariant. That is, the translation of the DLO without changes of the internal shape will not alter the Jacobian matrix. Thus, for the input of the NN, we represent the position of each feature by its relative position to its left adjacent feature (or the left end for the leftmost feature). It seems that the Jacobian is also rotation invariant, but it is only valid for rotations around the vertical axis because of gravity. We consider it by using a rotation data augmentation introduced in the next section.

If the model trained on one DLO is applied on another DLO of a very different length, the changed value range of the state input will make the model (NN) almost completely fail. Considering the adaptiveness on different DLOs, we propose the scale normalization. It is based on an approximation that there are similarities between the Jacobian matrices of DLOs with different lengths but similar shapes, which we call the approximate scale invariance. Fig. 3 is an illustration of the ideal cases, where two DLOs are moved from one identical overall shape to another identical overall shape, and the long DLO is \( \lambda L \) times as long as the short DLO. First, consider the translation of the grasped ends, as shown in Fig. 3(a). For the short DLO, the grasped end moves \( \delta p \), and the \( k \)th feature moves \( \delta \hat{x}_k \); for the long DLO, the grasped end moves \( \lambda L \delta p \), and the feature moves \( \lambda L \delta \hat{x}_k \). Next, consider the rotation of the grasped ends, as shown in Fig. 3(b). For the short DLO, the grasped end rotates \( \delta q \), and the feature moves \( \delta \hat{q} \); for the long DLO, the grasped end also rotates \( \delta q \), but the feature moves \( \lambda L \delta \hat{x}_k \). In these ideal cases, the proportional relationship between the translation of the end and the movement of the feature is independent of the scale, while that between the rotation of the end and the movement of the feature is proportional to the scale. Inspired by it, we define the approximate scale invariance as follows: for DLOs of different lengths but similar overall shapes, the Jacobian matrices for the linear velocities of the ends are similar, and those for the angular velocities of the ends are approximately proportional to the lengths.

Considering these properties, we split the Jacobian into two parts: an unknown matrix uncorrelated with the scale and translation and a constant scale matrix. The model (10) is modified to

\[
\dot{x}_k = J_k(s) \nu = (N_k(s)T) \nu, \quad k = 1, \ldots, m
\]

(11)

where \( s = [x; r] \), \( s \) is the relative state representation, and \( T \) is the scale matrix as

\[
T = \text{diag} \begin{bmatrix} I_{3 \times 3}, L I_{3 \times 3}, I_{3 \times 3}, L I_{3 \times 3} \end{bmatrix}
\]

(12)

where \( L \) is the length of the DLO. The relative state representation \( \hat{s} \) is specifically defined as

\[
\hat{s} := [\hat{x}_1; \cdots; \hat{x}_m; \hat{r}]
\]

(13)
where
\[ \ddot{x}_1 = \frac{x_1 - p_1}{\|x_1 - p_1\|}, \quad \ddot{x}_k = \frac{x_k - x_{k-1}}{\|x_k - x_{k-1}\|}, \quad k = 2, \ldots, m \]
\[ \hat{r} = \left[ \frac{p_2 - p_1}{\|p_2 - p_1\|}; q_1; q_2 \right] \]  \tag{14}

where the scale is normalized by the normalization of the relative position vectors.

In (11), only \( N_k(\tilde{s}) \) is unknown and will be approximated by an NN. The input of the NN is only related to the overall shape, ignoring the scale and translation. Therefore, it is much more data efficient than using the absolute state representation \([x; r]\), which requires a larger NN and more training data to guarantee the generalization to different DLO lengths and large translations.

Remark 1: The approximate scale invariance is only an approximation and may cause modeling errors. In fact, DLOs of the same length may also have different properties because of different materials and thicknesses. However, the approximation at least makes the offline-trained model able to work on DLOs of different lengths but not completely fail owing to the changed value range of the NN input. In the experiments, we demonstrate that this approximation is effective. Moreover, the remaining modeling errors can be compensated for by the online model adaptation on the specific DLO.

V. OFFLINE MODEL LEARNING

Prior to the shape control tasks, an initial approximation of the model by an NN is learned based on offline-collected random motion data. This section introduces the NN model, data collection method, and training details.

A. NN Model

We apply an RBFN to approximate \( N_k(\tilde{s}) \) in the Jacobian model. The actual \( N_k(\tilde{s}) \) is represented as
\[ \text{vec}(N_k(\tilde{s})) = W_k\theta(\tilde{s}), \quad k = 1, \ldots, m \]  \tag{15}
where \( W_k \) is the unknown actual RBFN weights for the \( k \)th feature and \( \theta(\tilde{s}) = [\theta_1(\tilde{s}), \theta_2(\tilde{s}), \ldots, \theta_g(\tilde{s})]^T \in \mathbb{R}^g \) is the vector of activation functions. We use the Gaussian radial function as the activation function
\[ \theta_i(\tilde{s}) = e^{-\frac{|x_i - \mu_i|^2}{\sigma_i^2}}, \quad i = 1, \ldots, g \]  \tag{16}
where \( g \) is the number of the hidden neurons and \( \mu_i \) and \( \sigma_i \) are the center and the width of the \( i \)th hidden neuron.

Equation (15) can be decomposed as
\[ N_{ki}(\tilde{s}) = W_{ki}\theta(\tilde{s}), \quad i = 1, \ldots, 12 \]  \tag{17}
where \( N_{ki} \) is the \( i \)th column of \( N_k \), and \( W_{ki} \) is the \((3(i-1) + 1)\)th to \((3i)\)th rows of \( W_k \). Substituting (17) into (11) yields
\[ \dot{x}_k = N_k(\tilde{s})T\nu = \sum_{i=1}^{12} W_{ki}\theta(\tilde{s})T_i\nu_i = \sum_{i=1}^{12} W_{ki}\theta(\tilde{s})T_i\nu_i \]  \tag{18}
In large deformation control tasks, there may be large translations or rotations. Our model using the relative state representation is natively translation invariant. To achieve rotation invariance around the vertical axis, we introduce a rotation data augmentation, which is inspired by observing the motion of the DLO and end-effectors in another coordinate that is defined by rotating the original world coordinate $W$ around the vertical axis. The data are transformed from $W$ to $R$ and then sent to the NN for training.

After applying the rotation data augmentation, our model is both translation-invariant and vertical-rotation-invariant. As a result, our model can handle large translations and rotations, with no need to collect more data to guarantee generalization. It complements our data collection method, which is convenient to implement but restricts the moving range of the DLO. We also find that it effectively reduces the overfitting of the NN when the collected dataset is small, since infinite new data can be generated. This is why we do not directly consider it in the relative state representation.

2) Domain Randomization on Different DLOs: To improve the generalization ability of the offline learned model on different DLOs, we apply a domain randomization method during the offline learning. That is, we train the offline model based on the combined data of several different DLOs with different lengths and thicknesses. This is for learning an offline model, which is an acceptable initial estimation for different DLOs. Then, for any new DLO in the manipulation, this model can be efficiently updated via the online adaptation.

Note that when using the domain randomization, the proposed scale normalization is still meaningful and effective, since it reveals the similarities between DLOs of different lengths and ensures that the ranges of the input values for the NN are consistent, which reduces the learning difficulty.

VI. SHAPE CONTROL WITH ONLINE MODEL ADAPTATION

A. General Control Problem Formulation

The control objective is to move the target points on the DLO to the desired positions. The target points can be any subset of the features, whose indexes form set $C$. Then, the shape vector, Jacobian matrix, and prediction error vector for the target points are denoted as

$$x^c = \begin{bmatrix} x_k^c \end{bmatrix}, \quad J^c(s) = \begin{bmatrix} J_k(s) \end{bmatrix}, \quad e^c = \begin{bmatrix} e_k \end{bmatrix}, \quad k \in C.$$  \hfill (22)

We denote the task error as $\Delta x^c = x^c - x_{\text{des}}^c$, where $x_{\text{des}}^c$ is the desired position vector of the target points. The velocities of the end-effectors $\nu$ are kinematically controlled.

Generally, the task can be formulated as an optimal control problem, in which the objective is to manipulate the DLO to the desired shape in a shorter time with smaller end-effector velocities while satisfying the valid-state constraints. However, since the system model is nonlinear and approximate and the constraints may be complex, the exact solving of the optimal control problem is impossible. An alternative approach is to apply MPC instead. For such a nonlinear and complex problem, sampling-based MPC methods, such as cross-entropy method and model-predictive path integral (MPPI) [43], are usually...
used. To apply them, the estimated state equation needs to be approximately discretized as

$$x(t + \delta t) = x(t) + \mathbf{J}(s(t)) \nu(t) \delta t$$  \hspace{1cm} (23)$$

where $\delta t$ is the time step interval. Then, the control problem is formulated as

$$\min_{\nu(t+i\delta t)} J_M = \frac{1}{2} \| \Delta x^c(t+T_h \delta t) \|_2^2 + \frac{\lambda}{2} \sum_{i=0}^{T_h-1} \| \nu(t+i\delta t) \|_2^2$$

$$0 \leq i < T_h$$

s.t. $\| \nu(t+i\delta t) \|_2^2 \leq \nu_{\text{max}}^2$,  \hspace{1cm} 0 \leq i < T_h

$$s(t+i\delta t) \in S_{\text{valid}}, \hspace{1cm} 0 \leq i \leq T_h$$  \hspace{1cm} (24)$$

where $t$ is the current time, $T_h$ is the planning horizon, $\nu_{\text{max}}$ is the maximum allowed end-effector speed, and $S_{\text{valid}}$ is the set of valid states. This optimization problem is solved every step to update the control inputs.

The conventional MPC commonly requires an accurate model and may not be able to deal with huge model errors when applied on new DLOs. Moreover, the sampling-based MPC is usually computationally intensive because they require large amounts of sampling sequences and NN-based predictions for future states. Therefore, we refer to the concept of MPC and propose an optimization-based adaptive controller, which can efficiently calculate the control input in the presence of an inaccurate model, handle various constraints, and further update the model online to adapt to new DLOs.

### B. Adaptive Controller Through Online Model Adaptation

We propose an adaptive controller to achieve the adaptivity to new DLOs through online model adaptation, whose structure is illustrated in Fig. 6. During the manipulation, the offline-learned RBFN is further updated to compensate for the modeling errors. The control input is efficiently calculated by solving a convex optimization problem, which considers the singularity of the Jacobian and constrains the robots not to overstretched the DLO. The stability of the closed-loop system is theoretically guaranteed.

1) Control Law: First, we define a saturated task error as

$$\Delta \hat{x}^c = \begin{cases} \Delta x^c, & \| \Delta x^c \|_2 \leq \epsilon_s \\ \frac{\epsilon_s}{\| \Delta x^c \|_2} \Delta x^c, & \text{otherwise} \end{cases}$$  \hspace{1cm} (25)$$

where $\epsilon_s$ is the saturation threshold. Then, we define an ideal velocity vector of the target points as

$$\hat{x}_{\text{ide}}^c = -\alpha \Delta \hat{x}^c$$  \hspace{1cm} (26)$$

where $\alpha$ is a positive gain factor. The ideal velocity vector of the target points is in the opposite direction of the task error $\Delta x^c$. It is obvious that the target points will converge to their desired positions $x^c_{\text{des}}$ if they move at $\hat{x}_{\text{ide}}^c$. However, $\hat{x}_{\text{ide}}^c$ is unachievable in most cases, since the tight coupling between different target points makes them unable to move in arbitrary directions. Thus, what we actually expect is to make the real velocities of the target points as close to the ideal velocities as possible. The benefit of the conversion from the task error to the ideal velocity vector as the control objective is that we can then formulate the controller as an optimization rather than just a feedback equation, in which essential constraints can be considered [44].

The control input is specified as the optimal solution of the convex optimization problem

$$\min_{\nu} J_A = \frac{1}{2} \| \hat{x}_{\text{ide}}^c - \hat{J}(s)\nu \|_2^2 + \frac{\lambda}{2} \| \nu \|_2^2$$

s.t. $\| \nu \|_2^2 \leq \nu_{\text{max}}^2$

$$C_1(s)\nu \leq 0$$

$$C_2(s)\nu = 0$$  \hspace{1cm} (27)$$

where the linear constraints are for avoiding overstretching the DLO, which is introduced in the next section. We can also constrain the DoFs of the end-effectors for specific tasks. Such a quadratically constrained quadratic program can be efficiently solved using convex optimization solvers.

Note that the form of the cost function in (27) is similar to the damped least-squares method, which is widely used in robot inverse kinematics and control to address the singularity of the robot Jacobian [45]. Here, we utilize it to deal with the singularity of the Jacobian of the DLO, which is very common during large deformation. In addition, $\lambda$ is chosen as $\lambda = \lambda_0 \| \Delta x^c \|_2$, where $\lambda_0$ is a positive constant. This means that when the current shape is far from the desired shape, a larger $\lambda$ is preferred for addressing the singularity problem, but when the current shape is near the desired shape, a smaller $\lambda$ is preferred for more precise control.

2) Constraints for Avoiding Overstretching: In large deformation control tasks, it is important to ensure that the DLO will not be overstretched during large motion in the presence of inaccurate deformation models. To address this problem, we add linear constraints to the motion of the ends when the DLO is going to be overstretched, without using the estimated deformation model.

We define the near-overstretched states as all DLO features lying almost in a straight line, which is mathematically described as

$$\frac{(x_{k+1} - x_k) \cdot (x_k - x_{k-1})}{\| x_{k+1} - x_k \|_2 \| x_k - x_{k-1} \|_2} > 1 - \epsilon_s \hspace{1cm} \forall k = 2, \ldots, m - 1$$  \hspace{1cm} (28)$$
where $\epsilon_s$ is a small threshold. Denote the vector between the positions of the left and right ends as $p_d = p_2 - p_1$. First, consider the linear velocities of the two ends. The constraint is that the linear velocity of the right end projected on $p_d$ must be no more than that of the left end projected on $p_d$

$$p_d \cdot v_2 - p_d \cdot v_1 \leq 0.$$  

(29)

Then, consider the angular velocities of the two ends, whose effect on whether the DLO will be overstretched is much more difficult to model. Thus, we simply add a strong constraint that the angular velocities of the left and right ends equal to zero

$$w_1 = w_2 = 0.$$  

(30)

Therefore, the constraints are specified as follows:

$$[-p_d^T \ 0_{1 \times 3} \ p_d^T \ 0_{1 \times 3}] \nu \leq 0$$  

(31)

$$[0_{3 \times 3} \ I_{3 \times 3} \ 0_{3 \times 3} \ 0_{3 \times 3} \ I_{3 \times 3}] \nu = 0.$$  

(32)

Considering both the normal and near-overstretched states, in (27), we set $C_1(s) = 0$ and $C_2(s) = 0$ for normal states and $C_1(s) = C_l$ and $C_2(s) = C_a$ for near-overstretched states.

3) Online Model Adaptation: Modeling errors may exist because of insufficient offline training or different properties between the manipulated DLO and the trained DLOs. Thus, we further update the model migrated from the offline phase while carrying out the shape control task.

We maintain a sliding window of length $T_w$ to store the recent motion data whose timestamps form set $T_w$:

$$T_w = \{ \tau | \tau = t - i \delta t, 0 \leq i < T_w \}$$  

(33)

where $t$ is the current time.

The online updating law of the $j$th row of $W_{kj}$ of the RBFN is specified as

$$\dot{W}_{kj}^T(t) = \eta \left[ \theta(\ddot{s}(t)) T_i v_i(t) \Delta \ddot{x}_{kj}(t) \right.$$  

$$+ \gamma T_w \sum_{\tau \in T_w} \theta(\ddot{s}(\tau)) T_i \frac{\mu_k(\tau)}{n_v(\tau)} e_{kj}(\tau, t) \left] \right.$$  

(34)

where $\Delta \ddot{x}_{kj}$ is the $j$th element of the saturated task error $\Delta \ddot{x}$, and $e_{kj}$ is the $j$th element of the prediction error $e_k$. Note that $e_k(\tau, t)$ represents the prediction error of the stored data at time $\tau$ using the updated estimated Jacobian model at time $t$. In addition, $\eta$ is the positive online learning rate, $\gamma$ is a positive weight coefficient, and $n_v(\tau)$ is a normalization factor specified as

$$n_v(\tau) = \begin{cases} \| \ddot{x}(\tau) \|_2, & \| \ddot{x}(\tau) \|_2 \geq \epsilon_v \\ \epsilon_v, & \text{otherwise} \end{cases}$$  

(35)

where $\epsilon_v$ is a positive threshold to avoid amplifying the noise when the velocities are very small. Such updating is done for all $k \in C, i = 1, \ldots, 12$, and $j = 1, \ldots, 3$.

The proposed online updating law (34) has several advantages: 1) considering all recent data in the sliding window instead of only the latest data can reduce the effect of sensing noise; 2) the updating is driven by both the task error and the prediction error, which enables faster and more stable updating; 3) the combination of the model updating law and control law theoretically guarantees the stability of the closed-loop system; and 4) the online updating starts from the pretrained results obtained in the offline learning stage, so that the advantages of the learning in both phases are fully explored and combined.

**Theorem 2:** When the adaptive control scheme described by (27) and (34) is applied to the robot system for the shape control of DLOs, the closed-loop system is stable and the task error $\Delta \ddot{x}$ is bounded in the presence of modeling errors. Furthermore, the task error will converge to zero as $t \to \infty$, unless the prediction errors $e_k(\tau, t)$ for all data in the sliding window are zero as well as the optimal solution of (27) is zero at a configuration on the path.

**Proof:** See Appendix A.

**Remark 3:** The proposed control method assumes that the desired positions are achievable. Thus, the desired shapes are set as prerecorded shapes of the manipulated DLO in our experiments. In future works, we will study the determination of achievable desired shapes for specific tasks from the perspective of planning.

VII. SIMULATION RESULTS

A. Evaluation Metrics

The evaluation metrics used in the simulation and also the real-world experiments are introduced as follows.

1) Deformation Magnitude: We divide the generalized deformation of a DLO between time $t_1$ and $t_2$ into two parts: translation and relative deformation. The translation refers to the translation of the centroid of the DLO (approximated by the average position of all features), which is specified as

$$D_t(x(t_1), x(t_2)) = \| \bar{x}(t_1) - \bar{x}(t_2) \|_2$$  

(36)

where $\bar{x} = \frac{1}{m} \sum_{k=1}^{m} x_k$. Then, the relative deformation is defined as the average of the movement of each feature relative to the centroid, which is specified as

$$D_{rd}(x(t_1), x(t_2))$$  

$$= \frac{1}{m} \sum_{k=1}^{m} \| (x_k(t_1) - \bar{x}(t_1)) - (x_k(t_2) - \bar{x}(t_2)) \|_2.$$  

(37)

Note that the relative deformation describes changes of the overall shape while ignoring translations.

2) Metrics for Modeling Accuracy: The shape prediction error is defined as

$$\epsilon_{\text{shape}} = \| x_{\text{pred}} - x_{\text{groundtruth}} \|_2.$$  

(38)
The relative prediction error of the feature velocity vector using our Jacobian model is defined as

\[
e_{\text{vel}} = \frac{\|\dot{x} - \dot{J}(s)v\|_2}{\|\dot{x}\|_2} \times 100\%.
\] (39)

3) Metrics for Shape Control: Criteria for evaluating shape control performance are as follows.

a) Final task error: It is the final Euclidean distance between the desired position vector and final position vector within 30 s

\[e_{\text{control}} = \|\dot{x}^c(t_f) - \dot{x}^\text{des}\|_2, \quad t_f = 30 \text{ s}. \] (40)

b) Average task error of all the cases: It is the average final task error over all the cases.

c) Success rate: If the final task error is less than 5 cm, this case is regarded successful.

d) Average task error of successful cases: It is the average final task error over all the successful cases.

e) Average task time: It is the average time used to achieve success over all the successful cases. The task time is for reference only, since it depends on the control gain in servo methods or the bound of control inputs in MPC.

Note that the above \(e_{\text{shape}}, e_{\text{vel}}, \) and \(e_{\text{control}}\) contain the errors of all the features/target points without averaging.

B. Simulation Setup

The simulation environment is shown in Fig. 7. The simulation of DLOs is based on Obi [46], a unified particle physics engine for deformable objects in Unity3D [47]. Unity ML-Agents Toolkit [48] is used for the communication between Unity and Python scripts. The two ends of the DLO are grasped by two grippers, which can translate and rotate. The DLO shape is represented by eight features \((m = 8)\). The data collection frequency, control frequency, and online learning frequency are 10, 10, and 50 Hz, respectively \((\delta t = 0.1)\).

The hyperparameters for the controller are set as \(\alpha = 1.0, \lambda_0 = 0.1, \epsilon_x = 0.2, \) and \(\epsilon_a = 0.002\); those for the online model adaptation are set as \(T_w = 20, \gamma = 10, \epsilon_v = 0.01, \) and \(\eta = 1.0\).

C. Offline Learning of the Deformation Model

The offline data of DLOs are collected in simulation, using the method introduced in Section V-B. An RBFN with 256 hidden neurons \((q = 256)\) is first trained offline to learn the initial model. We perform a series of quantitative comparative studies to validate the proposed Jacobian model and offline training methods.

1) Our Jacobian Model Versus Forward Kinematics Model: We compare the offline modeling accuracy of the FKMs and our Jacobian model on a certain DLO, by using the trained models to predict the DLO shape after ten steps [the forward prediction by our Jacobian model is by (23)]. FKMs based on different network architectures are implemented as baselines, including the multilayer perceptron (MLP) [12], the bidirectional LSTM (biLSTM) [18], and the combination of the interaction network and biLSTM (IN-biLSTM) [19]. We also test the relationship between their modeling accuracy and the amount of training data. All training data and test data are from the same DLO. The error is the average Euclidean distance between the prediction and ground truth of the shape after ten steps. (a) 2-D. (b) 3-D.

Fig. 8. Comparison between our Jacobian model and the FKMs, and the relationship between the offline modeling accuracy and the amount of training data. All training data and test data are from the same DLO. The error is the average Euclidean distance between the prediction and ground truth of the shape after ten steps. (a) 2-D. (b) 3-D.
2) Model Generalization Improvement: We introduce the model modification considering the translation invariance and approximate scale invariance in Section IV-B and the rotation data augmentation in Section V-C1. All these are to improve the model’s generalization ability on different DLOs and different deformation shapes. To validate these methods, we design the following tests in 3-D scenarios, in which the evaluation criterion is the relative prediction error of the feature velocity vector as (39).

First, we validate the influence of the translation-invariant relative state representation and the rotation data augmentation on the data of a certain DLO. We train models using absolute state or relative state and using the translation and rotation data augmentation. Then, we respectively test the models on an original collected testset, and a testset with random translation and rotation transformation. From the results shown in Fig. 10, we find that: 1) both the relative state representation and rotation data augmentation contribute to the improvement of the modeling accuracy; 2) the rotation data augmentation can significantly reduce the model’s overfitting when the training dataset is small; and 3) the proposed methods make the model data-efficient and perform equally well on the original collected testset and randomly transformed testset.

Second, we validate the proposed scale normalization by training and testing the models on DLOs of different lengths. We collect data of 11 DLOs of different lengths and thicknesses (details are shown in Table I) and then compare the models with and without using the scale normalization in the following two experiments. First, we train the two models using 60k data of only DLO 0 and test them on 6k data of other DLOs, respectively. Second, we train the models on $10 \times 6k$ data of ten DLOs and test them on 6k data of the remaining one. The results in Fig. 11 indicate that: 1) when trained with only the data of DLO 0 of length 0.5 m, the model using the scale normalization directly generalizes well to other DLOs of lengths from 0.3 m to 1.2 m, while the model not using it performs terribly; and 2) when trained with the combined data of different DLOs, the model using the scale normalization still outperforms the other one. The results demonstrate that the approximate scale invariance is reasonable, and the proposed scale normalization is effective.

D. Shape Control With Online Model Adaptation

We detailedly analyze the proposed method for DLO shape control in 3-D simulation tasks and compare it with the existing methods in both the 2-D and 3-D tasks. We conduct all the tasks...
on DLO 0, in which 100 different feasible desired shapes are randomly chosen for testing. In most cases, there is a large deformation between the desired shape and the initial shape. All the DLO features are set as the target points. The performance criteria are introduced in Section VII-A3.

1) Effect of Online Model Adaptation: In Section VI-B, we propose an adaptive controller through online model updating to compensate for the offline modeling error owing to insufficient offline training or changes of the DLO properties. We validate the effect of the online adaptation for different initial offline models and with different online learning rates.

We first train six models based on different offline data, in which three are trained on 2k/10k/60k data from the DLO 0 (the manipulated DLO), and the other three are trained on 10×0.2k/10×1k/10×6k data from the other ten DLOs. The control performances of three settings are compared: 1) directly using the offline models trained on the same DLO; 2) directly using the offline models trained on different DLOs; and 3) using the offline models trained on different DLOs and using the online adaptation (online learning rate $\eta = 1.0$). For each setting, the models using different amounts of offline data are tested. The results shown in Fig. 12 indicate that: 1) the models trained on more offline data achieve better control performance; 2) the models trained on the same DLO are better than those trained on different DLOs; and 3) the online adaptation can effectively compensate for the effect of insufficient offline data or different properties between the trained DLOs and the manipulated DLO.

We further test the method’s sensitivity to the online learning rate $\eta$, as shown in Fig. 13. It indicates that the online adaptation performs relatively stably with a large range of learning rates from $10^{-2}$ to $10^3$. We finally choose $\eta = 1.0$ as the most proper learning rate and use it in other simulation and real-world experiments.

2) Comparison Between Different Models and Controllers: We further analyze the effect of different models (FKM versus our Jacobian model) and different controllers (MPC versus our controller) in shape control. The MPPI [43] is used as the specific MPC. Both the models are offline trained on the same $10 \times 6k$ data of ten DLOs different from the manipulated DLO. For a fair comparison, the online adaptation is not executed.

First, we compare the two models by using the same MPC. Then, we compare the different controllers by using the same Jacobian model, in which we also test a naive P controller specified as

$$\nu = -\alpha \left( \hat{J}^c(s) \right)^\top \Delta \tilde{x}^c. \quad (41)$$

The results in Table II show that: 1) using the same MPC, our Jacobian model greatly outperforms the FKM; 2) using the same Jacobian model, our controller achieves a success rate similar to the MPC’s but higher control accuracy; and 3) the naive P controller performs poorly because of the possible singularity problem of the Jacobian matrix.

3) Comparison With Existing Methods: We choose two representative classes of existing methods for comparison: 1) the offline method: learning an FKM of DLOs offline and using the MPC for shape control (FKM+MPC); and 2) the online method: estimating the Jacobian matrix online using weighted least-squares estimation (WLS) and using the same control law as (27). Compared with them, our method benefits from both the offline learning and online adaptation. All the offline models are trained on the data from the other ten DLOs. Both the 2-D and 3-D tasks are tested.

As shown in Table III, our method significantly outperforms the compared methods on both the success rate and the average task error. Compared with the online WLS, our method performs better using only $10 \times 0.2k$ (3.3 min) offline data and much better using more offline data. Since the WLS online estimates a local model, which only utilizes the limited local data, its performance on cases with large deformation is poor. Besides, it costs the longest time since it needs to initialize the Jacobian by moving the DLO ends in each DoF every time it starts. Compared with the offline FKM+MPC, our method performs significantly better using the same $10 \times 6k$ (100 min) offline data and performs comparably using only 1/30 of the data. The poor performance of the FKM+MPC is due to the FKM’s lower offline modeling accuracy and the lack of further updating for the untrained manipulated DLO, which also causes its highest average task error over the successful cases. We visualize some of the tasks accomplished using our method in Fig. 7.
TABLE II
COMPARISON BETWEEN DIFFERENT MODELS AND CONTROLLERS IN 3-D SHAPE CONTROL TASKS IN THE SIMULATION

| Model\(^a\) | Controller\(^b\) | Average task error of all cases (cm) ↓ | Success rate ↑ | Average task error of successful cases (cm) ↓ | Average task time of successful cases (s) ↓ |
|-------------|----------------|---------------------------------------|-----------------|---------------------------------------------|----------------------------------------|
| FKM         | MPC            | 5.780                                 | 69/100          | 2.035                                       | 12.371                                 |
| Our Jacobian model | MPC          | 3.588                                 | 81/100          | 1.556                                       | 9.567                                  |
| Our Jacobian model | Naive P controller | 12.163                               | 52/100          | 0.380                                       | 10.662                                 |
| Our Jacobian model | Our controller | 2.680                                 | 80/100          | 0.422                                       | 5.739                                  |

\(^a\) All the offline models are trained with the same 10 × 6k data of DLOs 1–10.
\(^b\) The online model adaptation of our method is not executed for a fair comparison.

TABLE III
COMPARISON WITH THE EXISTING METHODS IN 2-D AND 3-D DLO SHAPE CONTROL TASKS IN THE SIMULATION

| Scenario | Methods | Offline training data | Average task error of all cases (cm) ↓ | Success rate ↑ | Average task error of successful cases (cm) ↓ | Average task time of successful cases (s) ↓ |
|----------|---------|-----------------------|----------------------------------------|-----------------|---------------------------------------------|----------------------------------------|
| 2-D      | WLS     | -                     | 9.016                                  | 63/100          | 0.186                                       | 6\(^*\) + 5.500                         |
|          | FKM+MPC | 10 × 6k               | 7.944                                  | 54/100          | 1.961                                       | 10.961                                 |
|          | Ours    | 10 × 0.2k             | 1.175                                  | 90/100          | \(\underline{0.013}\)                        | 5.370                                  |
|          | Ours    | 10 × 6k               | 0.185                                  | 95/100          | \(\underline{0.013}\)                        | 5.397                                  |
| 3-D      | FKM+MPC | 10 × 6k               | 9.133                                  | 64/100          | 0.953                                       | 12\(^*\) + 7.317                        |
|          | Ours    | 10 × 0.2k             | 5.780                                  | 69/100          | 2.035                                       | 12.371                                 |
|          | Ours    | 10 × 6k               | 4.996                                  | 69/100          | 0.626                                       | 10.793                                 |
|          | Ours    | 10 × 6k               | 1.223                                  | 94/100          | \(\underline{0.175}\)                        | 5.777                                  |

\(^b\) Time for initializing the Jacobian matrix estimation.

VIII. REAL-WORLD EXPERIMENT RESULTS

A. Experiment Setup

The experiment setup is shown in Fig. 14. The two ends of the DLO are rigidly grasped by dual UR5 robots. The positions and velocities of the DLO features are obtained by applying Kalman filters on the measurement of the positions of the red markers. For the 2-D tasks, the DLO is placed on a table and tracked by the top RGB camera; for the 3-D tasks, the DLO is tracked by the front structured-light RGBD camera.

Fig. 14. Setup of the real-world experiments. The DLO is rigidly grasped by the two robot manipulators. For the 2-D tasks, the DLO is placed on the table and tracked by the top RGB camera; for the 3-D tasks, the DLO is tracked by the front structured-light RGBD camera.

B. Comparison With Existing Methods on Various DLOs

We validate the proposed method on several different DLOs with different materials, lengths, and thicknesses in the real world and compare it with the WLS and FKM+MPC. The used DLOs and their detailed parameters are shown in Fig. 15. In the 2-D tests, DLOs 1–4 are used, while in the 3-D tests, DLO 5 is used instead of DLO 4, because DLO 4 cannot be sensed precisely by the structured-light camera since it is black and very thin. We conduct three tests with different feasible desired shapes on each DLO. All the used offline models are trained using \(10 \times 6k\) data of ten DLOs in the simulation, which means no real-world data are collected for offline training. All the DLO features are set as the target points. For the WLS, we reduce the control gain \(\alpha\) from 0.3 to 0.1 and the control frequency to 1 Hz, because otherwise it performs significantly unstable and devices is based on ROS [49]. Limited by the maximum frame rate of the RGBD camera, the control frequency is set as 5 Hz (\(\delta t = 0.2\)).

The hyperparameters for the controller are set as \(\alpha = 0.3, \epsilon_x = 0.02, \) and \(\lambda_0 = 0.3/1.0\) for the 2-D/3-D tasks; those for the online model adaptation are set as \(T_w = 10, \gamma = 10, \epsilon_v = 0.01, \) and \(\eta = 1.0.\)

Fig. 15. DLOs used in the real-world experiments and their parameters.
TABLE IV
COMPARISON WITH THE EXISTING METHODS IN 2-D AND 3-D DLO SHAPE CONTROL TASKS IN THE REAL-WORLD EXPERIMENTS

| Scenario | Methods | Average task error of all cases (cm) | Success rate | Average task error of successful cases (cm) | Average task time of successful cases (s) |
|----------|---------|-------------------------------------|--------------|--------------------------------------------|------------------------------------------|
| 2-D      | WLS     | 5.328                               | 9/12         | 2.146                                      | 12<sup>a</sup> 39.522                   |
|          | FKM+MPC | 3.000                               | 13/12        | 1.673                                      | 13 527                                   |
|          | Ours (w/o online adaptation) | 1.082 | 12/12 | 1.082                                      | 12 433                                   |
|          | Ours    | 0.475                               | 12/12        | 0.475                                      | 11 767                                   |
| 3-D      | WLS     | 7.475                               | 4/12         | 3.461                                      | 24<sup>b</sup> 63.2                     |
|          | FKM+MPC | 3.584                               | 9/12         | 2.911                                      | 14 311                                   |
|          | Ours (w/o online adaptation) | 1.680 | 11/12 | 1.318                                      | 11 491                                   |
|          | Ours    | 0.757                               | 12/12        | 0.757                                      | 9 283                                    |

<sup>a</sup> Time for initializing the Jacobian matrix estimation.

Unsmooth. For a fair comparison, we run the WLS for 90 s on each test, while we run the other methods for 30 s.

The results are summarized in Table IV, and the control processes of four cases are shown in Fig. 16. With the help of the online model adaptation, our method accomplishes all 24 tasks in 2-D and 3-D scenarios and achieves the fastest and most precise control. The FKM+MPC accomplishes most of the tests but with larger task errors. The WLS has the lowest success rate and the highest task error, demonstrating that the WLS is unsuitable for the large deformation control of DLOs.

The results of this research indicate that our method is capable of achieving competitive performance and can be further optimized with online adaptation. When compared to existing methods, our approach offers improved precision and success rates. However, further research is needed to address the limitations highlighted in the study, such as the initialization of the Jacobian matrix and the impact of measurement errors.

### C. Cases of Study

1) **Avoiding Overstretching:** We design a case to validate the proposed constraints for avoiding overstretching the DLO. As shown in Fig. 19, we set an infeasible desired shape whose length is 1.3 times the length of the DLO. The results show that, if the constraint is not used, the DLO is overstretched and falls out of the gripper; in contrast, if the constraint is used, the DLO is manipulated to a shape as close to the infeasible desired shape as possible but not overstretched.

2) **Different Choices of Target Points:** We show that any subset of the features can be set as the target points in Fig. 20. It is found that the final shapes of choosing all eight features or only three features as the target points are similar, which indicates the coupling of the features. When choosing only one target point, there are infinite solutions, but the robots will achieve the task with minimal movements of the end-effectors.

3) **Application: Embedding the DLO Into Grooves:** We show two cases of the potential applications of the proposed method.
Fig. 17. 2-D shape control results using our method in the real-world experiments. Each picture shows a completed task. The blue points represent the tracking results of the red features (also represent the target points). The green+black circles represent the desired positions of the target points. In all the cases, the DLO starts from a straight line in the center of the camera’s field of view. (Refer to our video for the full control processes.)

Fig. 18. 3-D shape control results using our method in real-world experiments. Each picture shows a completed task: the bottom subpicture shows the front view captured by the RGBD camera, which is actually used for the shape control; the top subpicture shows the top view captured by an RGB camera, which is only for a better illustration of the 3-D DLO shapes to readers but not used for tracking or control. Note that the task error shown in the top view looks a little larger owing to the small calibration error between the two cameras. (Refer to our video for the full control processes.)

Fig. 19. Case to show the effect of the constraints for avoiding overstretching. The DLO is manipulated to an infeasible desired shape whose length is 1.3 times the length of the DLO.

Fig. 20. Case of different choices of the target points. From left to right: all features, three features, and one feature are chosen as the target points.

In the first case, we apply the method to embed the DLO into the grooves, as shown in Fig. 21. The first, fourth, fifth, and eighth features are set as the target points, and the three-DoF translation and the rotation around the Z-axis are allowed for each robot end-effector. Besides the final desired shape in the grooves, we also define an intermediate desired shape above the grooves. The DLO is first manipulated to an intermediate desired shape, and then the final desired shape, to complete the whole task.

4) Application: Hooking up a Hanger: In the second case, we apply the proposed method to control the DLO to hook up a hanger, which is initially hanging on a rope, as shown in Fig. 22. Only the fourth and fifth features are set as the target points. Only the translation along the X and Y axes is allowed for the robot end-effectors, which means that the robots can reach the hanger only by deforming the DLO, but not by just reducing the height of the end-effectors. The whole complicated task is divided into several parts, and an intermediate desired shape is defined for each part. These two cases also show that our method can be
A. Limitations

The limitations of our method include the following:

1) Our method is designed for slow (quasi-static) manipulations and cannot be applied in dynamic manipulations where the inertia should be considered, because the derivation of our Jacobian model is based on the quasi-static assumption.

2) This article focuses on control but not planning. As a result, the moving path of the DLO may not be globally optimal. In addition, collisions between robot arms and self-intersection of the DLO may happen. In more complicated tasks, a global planner should be introduced to roughly plan a proper path or necessary intermediate desired shapes ahead.

3) We cannot determine whether a desired shape is feasible before manipulation. In our method, if the desired shape is infeasible, the DLO will stop as close to the desired shape as possible, such as the case in Section VIII-C1 and Fig. 19. One possible way to roughly study the feasibility of a desired shape is to use an energy-based DLO model to decide whether the shape is at a local minimum of the deformation energy.

In addition, the perception of DLOs is a very challenging research topic; however, it is out of the scope of this article. In the real-world experiments, we simplify the sensing of DLO features by putting markers on the DLO and manually remove the cases where most methods fail owing to occlusions. Recently, some research works have preliminarily investigated marker-free perception approaches. These approaches track the virtual points along the DLO from the point cloud using the Gaussian mixture model and other constraints and can even handle slight occlusions [50], [51], [52]. These perception methods can be applied as the front end of our method in marker-free scenarios, in which the DLO features in our method are a subset of the virtual tracking points.

B. Conclusion

This article proposed a new scheme for the large deformation control of DLOs with coupled offline and online learning of the unknown global deformation model. The combination of offline and online learning enables both accurate global modeling and further updating for new DLOs during actual manipulation, which allows our method to handle large deformation tasks and adapt well to new DLOs. In the offline phase, an offline model is trained on random motion data of DLOs of different properties, to obtain an estimation with good generalization performance. Then, the offline model is seamlessly migrated to the online phase as an initial estimation. Finally, in the online phase, the shape control task is executed, while the model is concurrently updated based on online motion data to compensate for offline modeling errors.

In detail, we described the global deformation model by a nonlinear mapping from the DLO configuration to a local Jacobian matrix and proved its rationality. We also introduced several strategies to improve the model’s training efficiency and generalization ability, including the scale normalization and rotation data augmentation. As for the controller, we formulated it as the optimal solution of a convex optimization problem, which considers the singularity of the Jacobian matrix and constrains the robots not to overstretch the DLO. We used the Lyapunov method to analyze the stability and convergence of the whole system.

We conducted a series of simulations and real-world experiments to demonstrate that our method can stably and precisely achieve the large deformation control of DLOs and greatly outperforms the existing data-driven methods. We demonstrated that our Jacobian model is more data-efficient, and the online adaptation effectively compensates for offline model errors owing to insufficient training or changes of DLO properties. Using the offline model trained with only simulation data, our method accomplished all the 2-D and 3-D tasks on different DLOs in the real-world experiments with the highest accuracy and within roughly 10 s only.
In terms of future work, we would like to study the marker-free perception method to make our method more practical in reality.

We would also like to introduce a high-level planner, which can be combined with this control method to achieve complex tasks that require not only accurate final control results but also proper moving paths.

**APPENDIX A**

**PROOF OF THEOREM 2**

The stability of the system using the control law (27) and the online updating law (34) is analyzed as follows.

First, since \( \nu = 0 \) is a possible solution of the optimization problem (27), the optimal must be no more than \( \frac{1}{2} \parallel \hat{x}_{\text{ide}} \parallel^2 \). Thus, for the optimal solution \( \nu \), we have

\[
\frac{1}{2} \parallel \hat{x}_{\text{ide}} - \hat{J}^c(s) \nu \parallel^2 + \frac{\lambda}{2} \parallel \nu \parallel^2 \leq \frac{1}{2} \parallel \hat{x}_{\text{ide}} \parallel^2
\]

(42)

\[- (\hat{x}_{\text{ide}})^T \hat{J}^e(s) \nu + \frac{1}{2} (\nu)^T (\hat{J}^c(s))^T \hat{J}^c(s) \nu + \frac{\lambda}{2} (\nu)^T \nu \leq 0
\]

(43)

\[(\hat{x}_{\text{ide}})^T \hat{J}^e(s) \nu \geq \frac{1}{2} (\nu)^T (\hat{J}^c(s))^T \hat{J}^c(s) \nu + \frac{\lambda}{2} (\nu)^T \nu \geq 0.\]

(44)

Substituting (26) into it yields

\[(\Delta \hat{x}^c)^T \hat{J}^e(s) \nu \leq 0.\]

(45)

Next, substituting (21) and (22) into (11) and noticing that the desired position vector \( x_{\text{des}}^c \) is fixed, we have

\[\Delta \hat{x}^c = \hat{x}^c = \hat{J}^c(s) \nu
\]

\[= \hat{J}^c(s) \nu - \hat{J}^e(s) \nu + \hat{J}^e(s) \nu = \hat{J}^e(s) \nu + e^c.\]

(46)

Then, define a potential function of \( \Delta x^c \) as

\[P(\Delta x^c) = \begin{cases} \frac{1}{2} (\Delta x^c)^T \Delta x^c, \quad & \parallel \Delta x^c \parallel_2 < \epsilon_x \\ \epsilon_x \parallel \Delta x^c \parallel_2^2 - \frac{\epsilon^2_x}{2}, \quad & \text{otherwise} \end{cases}\]

(47)

and we have

\[
\frac{dP(\Delta x^c)}{d\Delta x^c} = \begin{cases} \Delta x^c, \quad & \parallel \Delta x^c \parallel_2 < \epsilon_x \\ \epsilon_x \frac{\Delta x^c}{\parallel \Delta x^c \parallel_2}, \quad & \text{otherwise} \end{cases} = \Delta \hat{x}^c.
\]

(48)

A Lyapunov-like candidate is given as

\[V = P(\Delta x^c) + \frac{1}{\eta} \sum_{k \in C} \sum_{i=1}^{12} \sum_{j=1}^{3} \Delta W_{kij} \Delta \dot{W}^T_{kij}\]

(49)

Differentiating (49) with respect to time:

\[
\dot{V} = \frac{dP(\Delta x^c)}{d(\Delta x^c)^T} \Delta \hat{x}^c + \frac{1}{\eta} \sum_{k \in C} \sum_{i=1}^{12} \sum_{j=1}^{3} \Delta W_{kij} \Delta \dot{W}^T_{kij}.
\]

(50)

Substituting (21), (34), (46), and (48), we obtain

\[
\dot{V} = \left( \Delta \hat{x}^c \right)^T \left( J^c(s) \nu + e^c \right) - \frac{1}{\eta} \sum_{k \in C} \sum_{i=1}^{12} \sum_{j=1}^{3} \Delta W_{kij} \Delta \dot{W}^T_{kij}
\]

\[= \left( \Delta \hat{x}^c \right)^T \hat{J}^e(s) \nu + \left( \Delta \hat{x}^c \right)^T e^c
\]

\[- \frac{1}{T_w} \sum_{k \in C} \sum_{i=1}^{12} \sum_{j=1}^{3} \Delta W_{kij} \theta(s) T_{\tau} \nu_k(\tau) \frac{e_{kj}(\tau, t)}{n_{\nu}(\tau)}.\]

(51)

Substituting (21) and (22), we have

\[
\dot{V} = \left( \Delta \hat{x}^c \right)^T \hat{J}^e(s) \nu + \left( \Delta \hat{x}^c \right)^T e^c - \left( e^c \right)^T \Delta \hat{x}^c
\]

\[- \frac{1}{T_w} \sum_{k \in C} \sum_{i=1}^{12} \sum_{j=1}^{3} \Delta W_{kij} \theta(s) T_{\tau} \nu_k(\tau) \frac{e_{kj}(\tau, t)}{n_{\nu}(\tau)}\]

\[= \left( \Delta \hat{x}^c \right)^T \hat{J}^e(s) \nu - \frac{1}{T_w} \sum_{\tau \in T_w} \left( e^c(\tau, t) \right)^T \frac{e^c(\tau, t)}{n_{\nu}(\tau)} \geq 0\]

(52)

Since

\[
\frac{1}{T_w} \sum_{\tau \in T_w} \left( e^c(\tau, t) \right)^T \frac{e^c(\tau, t)}{n_{\nu}(\tau)} = 0 \quad \text{and} \quad \left( \Delta \hat{x}^c \right)^T \hat{J}^e(s) \nu = 0.
\]

(53)

First, consider the first term, which will equal to zero only if the prediction error \( e^c(\tau, t) = 0 \) for all data in the sliding window \( \tau \in T_w \). Such situations are very rare before the task is completed, where the approximate model should be absolutely accurate and all the prediction errors should be zero.

Then, consider the second term, which is equivalent to \( \left( \hat{x}_{\text{ide}}^c \right)^T \hat{J}^e(s) \nu = 0 \) from (26). According to (44), we have \( \left( \hat{x}_{\text{ide}}^c \right)^T \hat{J}^e(s) \nu = 0 \) only if the optimal solution of the problem (27) is \( \nu = 0 \). While the task is not completed \( \Delta x^c \neq 0 \), such situations may happen when the system is trapped into a local minimum point. It means that there are huge conflicts between the current task errors of different target points, so the controller based on the current estimated model thinks that any local robot movement cannot reduce the overall task error at the current configuration. In such an underactuated system, the local minimum is theoretically inevitable, but the experimental results demonstrate that such huge conflicts happen rarely as long as the desired position vector is feasible.

Only when these two conditions are met at the same time will \( \dot{V} \) equal zero. From a practical point of view, this is almost
impossible. As a result, it is reasonable that $\dot{V} < 0$ always holds before the task is completed. Since $V$ is positive definite, $\dot{V}$ is negative definite, and $V \to \infty$ as $\|\Delta x^c\|_2 \to \infty$ from (49), the convergence of $\Delta x^c \to 0$ as $t \to \infty$ is ensured, following [53].

**APPENDIX B**

**ADDITIONAL RESULTS**

A. Snapshots of Full Control Process

Fig. 24 provides snapshots of a case in real-world 3-D experiments to better illustrate the manipulation process to readers. Refer to our video for the full control processes of other cases.

B. Manipulating Very Short or Very Long DLOs

Generally speaking, very short DLOs are easier to model and manipulate, since their motion is usually more like rigid objects. Very long DLOs will easily get out of the dual-arm workspace and the camera’s field of view. In addition, they may be more deformable, and their inertial effect may be greater, making the modeling and manipulation harder.

Fig. 25 shows simulated shape control on a very short DLO (0.1 m) and a very long DLO (2.0 m). It indicates that our method can be applied to DLOs whose lengths are beyond the range of the offline trained DLOs’ lengths (0.3–1.2 m).

C. Impact of Sensing Noise

We conduct simulation tests to quantitatively study the impact of the sensing error on the overall control performance. We add Gaussian noise whose distribution is $\mathcal{N}(0, \sigma^2)$ to each dimension of each feature’s position, with zero mean and the variance of $\sigma^2$. Hence, the noise increases when $\sigma$ becomes larger. The feature velocity is simply calculated by $(x(t) - x(t - \delta t))/\delta t$, where $\delta t$ is the time step. Note that in this way, the feature velocity noise distribution is $\mathcal{N}(0, 2(\sigma/\delta t)^2)$. In our simulation, $\delta t$ is 0.1 s, so the standard deviation of the velocity noise is $10\sqrt{2}$ times that of the position noise. Other settings are the same as those in Section VII-D.

The test results are shown in Fig. 26. In our method, the Jacobian calculation and the feedback control law require the feature positions, and the online model adaptation is driven by both the feature positions and velocities. As the added noise increases, it is seen that: 1) in general, our method is robust to the noise, where the success rate changes little; 2) the average task error over successful cases increases, since the position noise causes oscillation around the desired shapes; and 3) the performance of the online model adaptation drops, since small position noise may cause very large velocity noise; to alleviate the issue, a front-end filter is preferred in noisy scenarios for more reliable velocity perception.

**REFERENCES**

[1] 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, 2018.

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

[3] F. Nadon, A. J. Valencia, and P. Payeur, “Multi-modal sensing and robotic manipulation of non-rigid objects: A survey,” *Robotics*, vol. 7, no. 4, 2018, Art. no. 74.

[4] R. Herguedas, G. López-Nicolás, R. Aragüés, and C. Sagüés, “Survey on multi-robot manipulation of deformable objects,” in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2019, pp. 977–984.
[5] X. Li, X. Su, Y. Gao, and Y. Liu, “Vision-based robotic grasping and manipulation of USB wires,” in Proc. IEEE Int. Conf. Robot. Autom., 2018, pp. 3482–3487.

[6] S. Jin, D. Romeres, A. Ragnathan, D. K. Jha, and M. Tomizuka, “Rejection-order optimization for manipulation of deformable objects: Assembly of belt drive units,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 10002–10008.

[7] L. Cao et al., “Sewing up the wounds: A robotic suturing system for flexible endoscopy,” IEEE Robot. Autom. Mag., vol. 27, no. 3, pp. 45–54, Sep. 2020.

[8] T. Lazerra, R. Gieselmann, F. T. Pokorny, and Y. Karayiannidis, “Reform: A robot learning sandbox for deformable linear object manipulation,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 4717–4723.

[9] H. Wakamatsu, E. Arai, and S. Hirai, “Knotting/unknotting manipulation of deformable linear objects,” Int. J. Robot. Res., vol. 25, no. 4, pp. 371–395, 2006.

[10] M. Saha and P. Isio, “Manipulation planning for deformable linear objects,” IEEE Trans. Robot., vol. 23, no. 6, pp. 1141–1150, Dec. 2007.

[11] D. McConachie, A. Dobson, M. Ruan, and D. Berenson, “Manipulating deformable objects by interleave planning, prediction, and control,” Int. J. Robot. Res., vol. 39, no. 8, pp. 957–982, 2020.

[12] P. Mitrano, D. McConachie, and D. Berenson, “Learning where to trust unreliable models in an unstructured world for deformable object manipulation,” Sci. Robot., vol. 6, no. 54, 2021, Art. no. eabi870.

[13] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, “Cable manipulation with a tactile-reactive gripper,” Int. J. Robot. Res., vol. 40, no. 12–14, pp. 1385–1401, 2021.

[14] J. Zhu et al., “Challenges and outlook in robotic manipulation of deformable objects,” IEEE Robot. Autom. Mag., early access, Feb. 25, 2022, doi: 10.1109/MRA.2022.3174415.

[15] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt, “Modeling of deformable objects for robotic manipulation: A tutorial and review,” Front. Robot. AI, vol. 7, 2020, Art. no. 82.

[16] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, “Learning predictive representations for deformable objects using contrastive estimation,” in Proc. 4th Conf. Robot Learn., 2020.

[17] W. Zhang, K. Schmeckpeper, P. Chaudhari, and K. Daniilidis, “Deformable linear object prediction using locally linear latent dynamics,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 13503–13509.

[18] M. Yan, Y. Zhu, N. Jin, and J. Bohg, “Self-supervised learning of state estimation for manipulating deformable linear objects,” IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 2372–2379, Apr. 2020.

[19] Y. Yang, J. A. Stork, and T. Stoyanov, “Learning to propagate interaction effects for modeling deformable linear objects dynamics,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 1950–1957.

[20] R. Lee, M. Hamaya, T. Murooka, Y. Ijiri, and P. Corke, “Sample-efficient learning of deformable linear manipulation in the real world through self-supervision,” IEEE Robot. Autom. Lett., vol. 7, no. 1, pp. 573–580, Jan. 2022.

[21] X. Lin, Y. Wang, J. Olkin, and D. Held, “SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation,” in Proc. 4th Conf. Robot Learn., 2020.

[22] R. Lazerra and Y. Karayiannidis, “Learning shape control of elastoplastic deformable linear objects,” in Proc. IEEE Int. Conf. Robot. Autom., 2021, pp. 4438–4444.

[23] D. Navarro-Alarcón, Y.-H. Liu, J. G. Romero, and P. Li, “Model-free visually servoed deformation control of elastic objects by robot manipulators,” IEEE Trans. Robot., vol. 29, no. 6, pp. 1457–1468, Dec. 2013.

[24] D. Navarro-Alarcón et al., “Automatic 3-D manipulation of soft objects by robotic arms with an adaptive deformation model,” IEEE Trans. Robot., vol. 32, no. 2, pp. 429–441, Apr. 2016.

[25] J. Zhu, B. Navarro, F. Fraise, A. Cronier, and A. Cherubini, “Dual-arm robotic manipulation of flexible cables,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2018, pp. 479–484.

[26] S. Jin, C. Wang, and M. Tomizuka, “Robust deformation model approximation for robotic cable manipulation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2019, pp. 6586–6593.

[27] R. Lagneau, A. Krupa, and M. Marchal, “Automatic shape control of deformable wires based on model-free visual servoing,” IEEE Robot. Autom. Lett., vol. 5, no. 4, pp. 2522–2529, Oct. 2020.

[28] J. Zhu, D. Navarro-Alarcón, R. Passama, and A. Cherubini, “Vision-based manipulation of deformable and rigid objects using subspace projections of 2D contours,” Robot. Autom. Syst., vol. 142, 2021, Art. no. 103799.
Mingrui Yu (Student Member, IEEE) received the B.Eng. degree in automation from Xi’an Jiaotong University, Xi’an, China, in 2020. He is currently working toward the Ph.D. degree in control science and engineering with the Department of Automation, Tsinghua University, Beijing, China. His research interests include robotic manipulation, especially for deformable objects.

Kangchen Lv received the B.Eng. degree in automation in 2021 from the Department of Automation, Tsinghua University, Beijing, China, where he is currently working toward the M.S. degree. His main research interests include robotic manipulation and perception.

Hanzhong Zhong received the B.Eng. degree in engineering mechanics in 2022 from the School of Aerospace Engineering, Tsinghua University, Beijing, China, where he is currently working toward the Ph.D. degree in control science and engineering with the Department of Automation. His research interests include robotic manipulation, multiagent systems, and deformable object modeling.

Shiji Song (Senior Member, IEEE) received the Ph.D. degree in mathematics from the Department of Mathematics, Harbin Institute of Technology, Harbin, China, in 1996. He is currently a Professor with the Department of Automation, Tsinghua University, Beijing, China. He has authored more than 180 research papers. His current research interests include pattern recognition, system modeling, optimization, and control.

Xiang Li (Member, IEEE) received the Ph.D. degree in robotics and automation from Nanyang Technological University, Singapore, in 2013. He is an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China. His current research interests include robotic manipulation, vision-based control, micro/nanorobots, and human–robot interaction. Mr. Li received the Highly Commended Paper Award from the International Federation for the Promotion of Mechanism and Machine Science in 2013, the Best Paper in Robotic Control at 2017 International Conference on Advanced Robotics, the Best Application Paper Finalists at 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, and the T. J. Tarn Best Paper in Robotics at 2018 IEEE International Conference on Robotics and Biomimetics. He was an Associate Editor for IEEE Robotics and Automation Magazine from 2019 to 2021 and International Conference on Robotics and Automation from 2019 to 2021. He has been an Associate Editor for IEEE ROBOTICS AND AUTOMATION LETTERS since 2022.