

# Shape Control of Deformable Linear Objects with Offline and Online Learning of Local Linear Deformation Models

Mingrui Yu, Hanzhong Zhong, and Xiang Li

**Abstract**—The shape control of deformable linear objects (DLOs) is challenging, since it is difficult to obtain the deformation models. Previous studies often approximate the models in purely offline or online ways. In this paper, we propose a scheme for the shape control of DLOs, where the unknown model is estimated with both offline and online learning. The model is formulated in a local linear format, and approximated by a neural network (NN). First, the NN is trained offline to provide a good initial estimation of the model, which can directly migrate to the online phase. Then, an adaptive controller is proposed to achieve the shape control tasks, in which the NN is further updated online to compensate for any errors in the offline model caused by insufficient training or changes of DLO properties. The simulation and real-world experiments show that the proposed method can precisely and efficiently accomplish the DLO shape control tasks, and adapt well to new and untrained DLOs.

## 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. For example, wires are manipulated to assemble devices in 3C manufacturing [1]; belts are manipulated in assemblies of belt drive units [2]; and in surgery, sutures are manipulated to hold tissue together [3].

The manipulation tasks of DLOs can be divided into two categories [4]. In the first category, the goals are not about the exact shapes of DLOs; rather, they concern high-level conditions such as insertion [5], tangling or untangling knots [6], obstacle-avoidance [7], flex and flip [8], etc. The second category is about manipulating DLOs to desired shapes, where one key challenge is to obtain the unknown deformation models, i.e., how the robot motion affects the DLO shapes. This paper focuses on the shape control tasks.

Different from rigid objects, it is challenging to obtain the exact models of DLOs, because they are hard to calculate theoretically, and may vary significantly among DLOs. Some analytical physics-based modeling methods can be used to model DLOs, such as mass-spring systems, position-based dynamics, and finite element methods [9]. However, all are approximate models, and require accurate parameters of

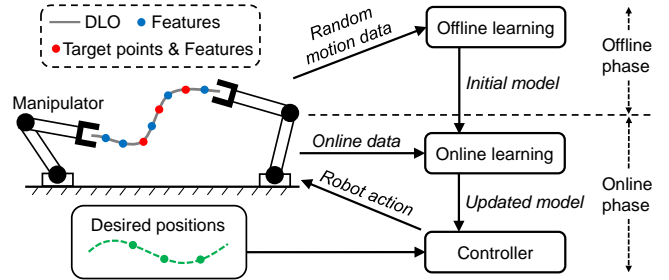


Fig. 1. Overview of the proposed scheme for DLO shape control. The shape of the DLO is represented by multiple features along the DLO. Some of the features are chosen as target points, and the task is defined as moving the target points to their desired positions. In the offline phase, an initial estimation of the deformation model is learned. Then, in the online phase, the shape control task is executed, and the model is further updated to compensate for offline modeling errors.

DLOs which are difficult to acquire. Data-driven approaches have been applied to learn the deformation models, without studying the complex physical dynamics. A common method is to first learn a forward kinematics model offline, and then use model predictive control in manipulation [10]–[13]. Although this allows for learning accurate forward kinematics of an offline-trained DLO, problems may arise when manipulating a different untrained DLO since there is no online update. Reinforcement learning methods have also been studied [14], [15], but they are less data-efficient, and the transfer from trained scenarios to untrained scenarios is challenging. Besides these offline approaches, some studies have used purely online methods to estimate the local linear deformation model of manipulated DLOs, which can be applied to any new DLO [16]–[19]. However, these online estimated models are less accurate because only a small amount of local data can be utilized.

In this paper, we propose a scheme for the shape control of DLOs, where the unknown deformation model is estimated with both offline and online learning, shown in Fig. 1. It allows more accurate modeling through offline learning and further updating for specific DLOs via online learning during manipulation. Specifically, we use a radial-basis-function neural network (RBFN) to model the mapping from the current state to the current local linear deformation model. In the offline phase, the RBFN is trained on the collected data. The offline model then directly migrates to the online phase as an initial estimation. In the online manipulation phase, an adaptive controller is proposed to control the shape, in which the RBFN is further updated to adapt to the manipulated DLO concurrently. Thus, the offline learning and online learning complement each other. In addition, we apply proper state representations and domain randomization methods

M. Yu, and X. Li are with the Department of Automation, Tsinghua University, China, and H. Zhong is with the School of Aerospace Engineering, Tsinghua University, China. This work was supported in part by the Science and Technology Innovation 2030-Key Project under Grant 2021ZD0201404, in part by the Institute for Guo Qiang, Tsinghua University, and in part by the National Natural Science Foundation of China under Grant U21A20517. Corresponding author: Xiang Li (xiangli@tsinghua.edu.cn)

to improve the model's generalization ability. The simple structure of RBFN and the linear format of the deformation model enable data-efficient offline training and fast online adaptation. Compared to the nonlinear offline methods, our method is more data-efficient and can adapt to untrained DLOs; compared to the purely online methods, our method is more stable and can handle more complex tasks. The stability of the closed-loop system and the convergence of task errors are analyzed using the Lyapunov method. Simulation and real-world experiment results are presented to demonstrate the better performance of the proposed scheme compared to the previous data-driven methods. The video and code are available at [https://mingrui-yu.github.io/shape\\_control\\_DLO/](https://mingrui-yu.github.io/shape_control_DLO/).

## II. RELATED WORK

In this section, existing approaches for the shape control of DLOs will be discussed.

Analytical physics-based modeling of DLOs has been researched over the past several decades [9]. Some works about shape control were based on physics-based models. In [20], finite element model (FEM) simulation of DLOs was used for open-loop shape control. An approach using reduced FEM to closed-loop shape control of DLOs was proposed in [21]. These methods highly rely on the accuracy of the analytical model, requiring the accurate DLO parameters which are hard to obtain in reality.

Data-driven approaches have been applied to the shape control of DLOs recently, dispensing with analytical modeling. In [22]–[24], the shaping of DLOs was addressed by learning from human demonstrations. Robots could reproduce human actions for specific tasks. Reinforcement learning (RL) has also been applied to learn policies for DLO shape control in an end-to-end manner. A simulated benchmark of RL algorithms for deformable object manipulation was presented in [14], in which the Soft Actor Critic (SAC) algorithm performs best in rope straightening tasks. RL policies for shape control of elastoplastic DLOs were learned in [15]. Like other RL applications, these methods suffer from high training expenses and challenging transfer from simulation to real-world scenarios.

Different from the end-to-end methods, many works first learn forward kinematics models of DLOs offline, and then use model predictive control (MPC) to control the shape. The forward kinematics model predicts the shape at the next time step based on the current shape and input action. In [10], [11], an encoder from the image space to the latent space, and a forward kinematics model 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 forward kinematics in the physical state space. A bi-directional LSTM network whose structure is similar to chain-like DLOs was applied in [12]. An interaction network was integrated with a bi-directional LSTM network in [13] to better learn the local interactions between segments of DLOs. The problem with these offline methods is that the generalization to different untrained DLOs cannot be guaranteed.

To control the shape of unknown objects, a series of methods tackle the shape control problem based on purely online estimation of the local linear deformation models of DLOs, in which a small change of the DLO is linearly related to a small displacement of the robot by a locally effective estimated Jacobian matrix. The control input can be directly calculated using the inverse of the Jacobian matrix. In [17]–[19], the local Jacobian matrix was obtained using the (weighted) least square estimation on only the data in the current sliding window. However, the accuracy of the online estimated models is limited and cannot be improved with more data. Thus, these methods mostly handle tasks with local and small deformation.

## III. METHODOLOGY

This paper considers the quasi-static shape control of elastic DLOs. ‘Quasi-static’ refers to the motion being slow, in which the shapes of DLOs are assumed to be determined by only their potential energies and no inertial effects [16]. As illustrated in Fig. 1, the robot end-effectors 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. The specific choice of the target points depends on the task needs.

Some frequently-used notations are listed as follows. The vertical concatenation of column vector  $\mathbf{a}$  and  $\mathbf{b}$  is denoted as  $[\mathbf{a}; \mathbf{b}]$ . The position vector of the end-effectors is represented as  $\mathbf{r} \in \mathbb{R}^n$ . The position of the  $i^{\text{th}}$  feature is represented as  $\mathbf{x}_i \in \mathbb{R}^l$ . The overall shape vector of the DLO is represented as  $\mathbf{x} = [\mathbf{x}_1; \dots; \mathbf{x}_m] \in \mathbb{R}^{lm}$ , where  $m$  is the number of the features. The dimension  $n$  and  $l$  are adjustable according to the requirements of the task.

### A. Local Linear Deformation Model

One key problem of the shape control of DLOs is to study the mapping from the motion of the end-effectors to the motion of the DLO features. The velocity vector of the DLO features can be locally linearly related to the velocity vector of the end-effectors using a Jacobian matrix [16]–[19]. Different from the previous works, we estimate the Jacobian matrix by learning the mapping from the current state  $(\mathbf{x}, \mathbf{r})$  to the current Jacobian matrix  $\mathbf{J}$ , i.e.,  $\mathbf{J}$  can be obtained as a function of  $\mathbf{x}$  and  $\mathbf{r}$ :

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{x}, \mathbf{r})\dot{\mathbf{r}} \quad (1)$$

*Proposition 1:* With 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 (1).

*Proof:* Denote the potential energy of the elastic DLO as  $E$ , which is assumed to be fully determined by  $\mathbf{x}$  and  $\mathbf{r}$ . In the quasi-static assumption, internal equilibrium holds at all states during the manipulation, where the DLO's internal shape  $\mathbf{x}$  locally minimizes the potential energy  $E$  [25]. That is,  $\partial E / \partial \mathbf{x} = \mathbf{0}$  at any state. Consider the DLO is

moved from state  $(\bar{x}, \bar{r})$  to state  $(\bar{x} + \delta x, \bar{r} + \delta r)$  where  $\delta x$  and  $\delta r$  are small displacements of the features and the end-effectors. Denote  $\partial E / \partial x$  as  $g(x, r)$ ,  $\partial^2 E / (\partial x \partial x)$  as  $A(x, r)$ , and  $\partial^2 E / (\partial x \partial r)$  as  $B(x, r)$ . Using Taylor expansion and neglecting higher order terms, we have

$$g(\bar{x} + \delta x, \bar{r} + \delta r) \approx g(\bar{x}, \bar{r}) + A(\bar{x}, \bar{r})\delta x + B(\bar{x}, \bar{r})\delta r \quad (2)$$

where  $g(\bar{x} + \delta x, \bar{r} + \delta r) = g(\bar{x}, \bar{r}) = 0$ . Note that  $A$  and  $B$  physically represent the unknown stiffness matrices. Assuming the DLO has a positive and full-rank stiffness matrix around the equilibrium point, matrix  $A$  is invertible [16]. Then, it can be obtained that

$$\delta x \approx -(A(\bar{x}, \bar{r}))^{-1} B(\bar{x}, \bar{r})\delta r \quad (3)$$

In slow manipulations,  $\dot{x} \approx \delta x / \delta t$  and  $\dot{r} \approx \delta r / \delta t$  with small  $\delta t$ . Then, denoting  $-(A(x, r))^{-1} B(x, r)$  as  $J(x, r)$ , we derive (1) and prove the proposition.

Note that (1) can be rewritten as

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_m \end{bmatrix} = \begin{bmatrix} J_1(x, r) \\ \vdots \\ J_m(x, r) \end{bmatrix} \dot{r} \quad (4)$$

where  $J_k(x, r)$  is the  $((k-1)l+1)^{\text{th}}$  to  $(kl)^{\text{th}}$  rows of  $J(x, r)$ . Thus, it can be obtained that

$$\dot{x}_k = J_k(x, r)\dot{r}, \quad k = 1, \dots, m \quad (5)$$

which indicates that different features correspond to different Jacobian matrices. This formulation makes it convenient when choosing any subset of features as the target points in the manipulation task.

However, it is difficult to theoretically calculate the Jacobian matrix. We estimate the Jacobian matrix in a data-driven way, combining both offline learning and online learning.

### B. Offline Learning

Prior to the shape control tasks, a data-driven learning method is employed to obtain the initial estimation of the model, based on offline collected data.

We apply a neural network (NN) to approximate the Jacobian matrix, in which the input is the current state and the output is the Jacobian. Two properties of the Jacobian can be noticed intuitively: (1) translation-invariance: translation of the whole DLO without changes of the shape will not alter the Jacobian matrix; (2) approximate scale-invariance: DLOs with different lengths but similar overall shapes and the same number of features may have similar Jacobian matrices. Thus, to improve the NN's generalization ability, we modify the representation of the input state from  $[x; r]$  to

$$\phi \triangleq [\bar{x}_1; \dots; \bar{x}_m; \bar{p}; q_0; q_1] \quad (6)$$

where

$$\bar{x}_k = \left[ \frac{x_k - p_0}{\|x_k - p_0\|}; \frac{x_k - p_1}{\|x_k - p_1\|} \right], \quad \bar{p} = \frac{p_1 - p_0}{\|p_1 - p_0\|} \quad (7)$$

( $k = 1, \dots, m$ ), where  $p_0, p_1$  are the positions of the left and right grasped ends of the DLO, and  $q_0, q_1$  are the

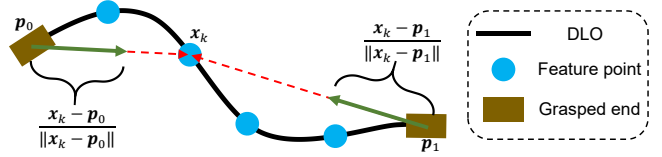


Fig. 2. An illustration of the proposed DLO state representation in (6). The position of the  $k^{\text{th}}$  feature  $x_k$  can be determined by  $\bar{x}_k = [(x_k - p_0) / \|x_k - p_0\|; (x_k - p_1) / \|x_k - p_1\|]$ , if the positions of the left and right ends ( $p_0$  and  $p_1$ ) are given. In (6), the positions of the ends are described by  $\bar{p} = (p_1 - p_0) / \|p_1 - p_0\|$ , so the overall translation and scale are ignored in the representation.

orientations of the left and right grasped ends. As illustrated in Fig. 2, this relative representation only determines the overall shape, ignoring the scale and overall translation. Therefore, it is much more data-efficient than the absolute representation  $[x; r]$  which requires a larger network and training dataset to guarantee the generalization to different DLOs. Note that this representation avoids using relative positions between adjacent features, which may suffer from accumulated errors when perception errors exist.

Then, (5) can be rewritten as

$$\dot{x}_k = J_k(\phi)\dot{r}, \quad k = 1, \dots, m \quad (8)$$

We apply a radial-basis-function neural network (RBFN) to represent the actual Jacobian matrix as a function of  $\phi$ :

$$\text{vec}(J_k(\phi)) = W_k \theta(\phi), \quad k = 1, \dots, m \quad (9)$$

where  $\text{vec}(\cdot)$  refers to the column vectorization operator, and  $W_k$  is the matrix of unknown actual weights of the RBFN for the  $k^{\text{th}}$  feature. The  $\theta(\phi)$  represents the vector of activation functions, and  $\theta(\phi) = [\theta_1(\phi), \theta_2(\phi), \dots, \theta_q(\phi)]^T \in \mathbb{R}^q$ . We use gaussian radial function as the activation function:

$$\theta_i(\phi) = e^{-\frac{\|\phi - \mu_i\|^2}{\sigma_i^2}}, \quad i = 1, \dots, q \quad (10)$$

where the parameters  $\mu_i$  and  $\sigma_i$  are trainable in the offline phase but fixed in the online phase.

Equation (9) can be decomposed as

$$J_{ki}(\phi) = W_{ki} \theta(\phi), \quad i = 1, \dots, n \quad (11)$$

where  $J_{ki}$  is the  $i^{\text{th}}$  column of  $J_k$ , and  $W_{ki}$  is the  $((i-1)l+1)^{\text{th}}$  to  $(il)^{\text{th}}$  rows of  $W_k$ . Subscribing (11) into (8) yields

$$\dot{x}_k = J_k(\phi)\dot{r} = \sum_{i=1}^n J_{ki}(\phi)\dot{r}_i = \sum_{i=1}^n W_{ki} \theta(\phi)\dot{r}_i \quad (12)$$

where  $\dot{r}_i$  is the  $i^{\text{th}}$  element of  $\dot{r}$ .

The estimated Jacobian matrix is represented as

$$\text{vec}(\hat{J}_k(\phi)) = \hat{W}_k \theta(\phi) \quad (13)$$

where  $\hat{W}$  is the matrix of estimated weights. The approximation error for the  $k^{\text{th}}$  feature  $e_k$  is specified as

$$\begin{aligned} e_k &= \dot{x}_k - \hat{J}_k(\phi)\dot{r} \\ &= \sum_{i=1}^n W_{ki} \theta(\phi)\dot{r}_i - \sum_{i=1}^n \hat{W}_{ki} \theta(\phi)\dot{r}_i = \sum_{i=1}^n \Delta W_{ki} \theta(\phi)\dot{r}_i \end{aligned} \quad (14)$$

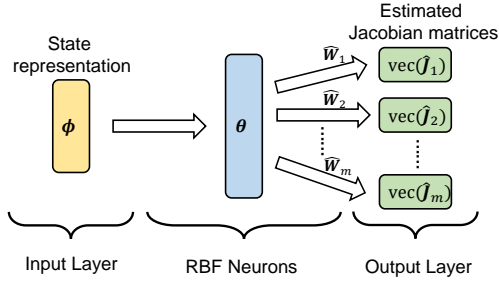


Fig. 3. The architecture of the RBFN for learning the local linear deformation model. The network takes the state representation in (6) as the input and outputs the estimated Jacobian matrices which relate the velocity vectors of DLO features to the velocity vector of the robot end-effectors.

The architecture of the RBFN is shown in Fig. 3. Note that the learning or estimation for Jacobian matrices of different features is carried out in parallel. In the offline phase, the ends of the DLO are controlled to move randomly to collect the training dataset, which contains  $\mathbf{x}_k, \dot{\mathbf{x}}_k, \mathbf{r}, \dot{\mathbf{r}}, (k = 1, \dots, m)$ . Then, the RBFN is trained on the collected data. Considering the noise and outliers in the data, we use the smooth  $L1$  loss [26] of  $\mathbf{e}_k$  for training.

The k-means clustering on a subset of the training data is used to calculate the initial value of  $\mu_i$  and  $\sigma_i, (i = 1, \dots, q)$ . Then, all parameters including  $\mu_i, \sigma_i$  and  $\hat{\mathbf{W}}$  are updated using the *Adam* optimizer [27]. We choose RBFN for its simple structure, robustness, and online learning ability [28]. Though less expressive than some more complex network architectures, it performs well enough in this work.

### C. Adaptive Control through Online Learning

Considering the differences between the manipulated DLO in the online phase and the trained DLOs in the offline phase, online learning during manipulation is required. We propose an adaptive control scheme, in which the offline estimated model is treated as an initial approximation and then further updated during the shape control tasks.

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  $\mathcal{C}$ . Then, the target shape vector  $\mathbf{x}^c$ , target Jacobian matrix  $\mathbf{J}^c(\phi)$ , and target weights  $\mathbf{W}^c$  are denoted as

$$\mathbf{x}^c \triangleq \begin{bmatrix} \vdots \\ \mathbf{x}_k \\ \vdots \end{bmatrix}, \mathbf{J}^c(\phi) \triangleq \begin{bmatrix} \vdots \\ \mathbf{J}_k(\phi) \\ \vdots \end{bmatrix}, \mathbf{W}^c \triangleq \begin{bmatrix} \vdots \\ \mathbf{W}_k \\ \vdots \end{bmatrix}, k \in \mathcal{C} \quad (15)$$

The velocities of the robot end-effectors  $\dot{\mathbf{r}}$  are controlled, and the control input is specified as

$$\dot{\mathbf{r}} = -\alpha (\hat{\mathbf{J}}^c(\phi))^\dagger \Delta \mathbf{x}^c \quad (16)$$

where  $(\hat{\mathbf{J}}^c(\phi))^\dagger$  is the Moore-Penrose pseudo-inverse of the estimated Jacobian matrix. In addition,  $\Delta \mathbf{x}^c = \mathbf{x}^c - \mathbf{x}_{desired}^c$  where  $\mathbf{x}_{desired}^c$  is the desired position vector of the target points, and  $\alpha \in \mathbb{R}$  is a positive control gain. In actual implementations,  $\dot{\mathbf{r}}$  is bounded to avoid too fast motion.

The online updating law of the  $j^{\text{th}}$  row of  $\hat{\mathbf{W}}_{ki}$  of the RBFN is specified as

$$\dot{\hat{\mathbf{W}}}_{kij}^T = \dot{\mathbf{r}}_i \theta(\phi) (\eta_1 \Delta x_{kj} + \eta_2 e_{kj}), \quad j = 1, \dots, l \quad (17)$$

where  $\Delta x_{kj}$  is the  $j^{\text{th}}$  element of the task error  $\Delta \mathbf{x}_k$ , and  $e_{kj}$  is the  $j^{\text{th}}$  element of the approximation error  $\mathbf{e}_k$ . The  $\eta_1$  and  $\eta_2$  are positive scalars. Such updating is done for all  $k \in \mathcal{C}$  and  $i = 1, \dots, n$ .

The proposed control scheme by (16) and (17) allows controlling the target points on the DLO to the desired positions while updating the RBFN concurrently to compensate for any offline modeling errors.

The stability of the system is analyzed as follows. Below  $\mathbf{J}^c(\phi), \mathbf{J}_k(\phi), \theta(\phi)$  are shortened to  $\mathbf{J}^c, \mathbf{J}_k, \theta$  for simplicity. Premultiplying both sides of (16) by  $\hat{\mathbf{J}}^c$ , we have

$$\hat{\mathbf{J}}^c \dot{\mathbf{r}} = -\alpha \hat{\mathbf{J}}^c (\hat{\mathbf{J}}^c)^\dagger \Delta \mathbf{x}^c \quad (18)$$

Note that from (14) and (15), it can be obtained that

$$\hat{\mathbf{J}}^c \dot{\mathbf{r}} = \hat{\mathbf{J}}^c \dot{\mathbf{r}} - \mathbf{J}^c \dot{\mathbf{r}} + \mathbf{J}^c \dot{\mathbf{r}} = -\mathbf{e}^c + \dot{\mathbf{x}}^c \quad (19)$$

where  $\mathbf{e}^c = [\dots; e_k; \dots], k \in \mathcal{C}$ . Since the desired positions are fixed, substituting (19) into (18) yields

$$\mathbf{e}^c = \Delta \dot{\mathbf{x}}^c + \alpha \hat{\mathbf{J}}^c (\hat{\mathbf{J}}^c)^\dagger \Delta \mathbf{x}^c \quad (20)$$

A Lyapunov-like candidate is given as

$$V = \frac{1}{2} (\Delta \mathbf{x}^c)^T \Delta \mathbf{x}^c + \frac{1}{2\eta_1} \sum_{k \in \mathcal{C}} \sum_{i=1}^n \sum_{j=1}^l \Delta \mathbf{W}_{kij} \Delta \mathbf{W}_{kij}^T \quad (21)$$

Differentiating (21) with respect to time and substituting (20) (17) and (14) into it, we can obtain that

$$\begin{aligned} \dot{V} &= (\Delta \mathbf{x}^c)^T \Delta \dot{\mathbf{x}}^c - \sum_{k \in \mathcal{C}} \sum_{i=1}^n \sum_{j=1}^l \frac{1}{\eta_1} \Delta \mathbf{W}_{kij} \dot{\mathbf{W}}_{kij}^T \\ &= -\alpha (\Delta \mathbf{x}^c)^T \hat{\mathbf{J}}^c (\hat{\mathbf{J}}^c)^\dagger \Delta \mathbf{x}^c + (\Delta \mathbf{x}^c)^T \mathbf{e}^c \\ &\quad - \sum_{k \in \mathcal{C}} \sum_{i=1}^n \sum_{j=1}^l \frac{1}{\eta_1} \Delta \mathbf{W}_{kij} [\dot{\mathbf{r}}_i \theta(\eta_1 \Delta x_{kj} + \eta_2 e_{kj})] \\ &= -\alpha (\Delta \mathbf{x}^c)^T \hat{\mathbf{J}}^c (\hat{\mathbf{J}}^c)^\dagger \Delta \mathbf{x}^c + (\Delta \mathbf{x}^c)^T \mathbf{e}^c \\ &\quad - (\mathbf{e}^c)^T \Delta \mathbf{x}^c - \frac{\eta_2}{\eta_1} (\mathbf{e}^c)^T \mathbf{e}^c \\ &= -\alpha (\Delta \mathbf{x}^c)^T \hat{\mathbf{J}}^c (\hat{\mathbf{J}}^c)^\dagger \Delta \mathbf{x}^c - \frac{\eta_2}{\eta_1} (\mathbf{e}^c)^T \mathbf{e}^c \leq 0 \end{aligned} \quad (22)$$

As  $V > 0$  and  $\dot{V} \leq 0$ , the closed-loop system is stable. The boundedness of  $V$  ensures the boundedness of  $\Delta \mathbf{x}^c$  from (21). If  $l \times |\mathcal{C}| \leq n$  and  $\hat{\mathbf{J}}^c$  holds full row rank,  $\hat{\mathbf{J}}^c (\hat{\mathbf{J}}^c)^\dagger$  is an identity matrix. Then, it can be proved that  $\Delta \mathbf{x}^c \rightarrow \mathbf{0}$  as  $t \rightarrow \infty$ , following [29]. Otherwise,  $\hat{\mathbf{J}}^c (\hat{\mathbf{J}}^c)^\dagger$  is only positive semi-definite, resulting in an underactuation system. However, from (14), (16) and (22) it can be proved that  $\dot{V} = 0$  if and only if  $\dot{\mathbf{r}} = \mathbf{0}$ , which only happens when there are huge conflicts between the desired moving directions of different target points so that the robot movement in any direction cannot reduce the  $\Delta \mathbf{x}^c$  at this "local minimum point". Actually, this situation happens rarely in experiments owing to the coupling between the target points, so in most cases  $\dot{V} < 0$  always holds and finally  $\Delta \mathbf{x}^c \rightarrow \mathbf{0}$ .



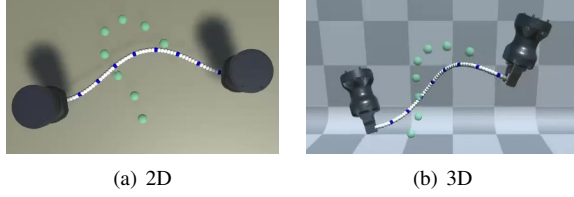


Fig. 4. The simulation environment for the DLO shape control tasks. The blue points represent the features along the DLO. The green points represent the desired positions of the features.

#### IV. RESULTS

We carry out both simulation and real-world experiments to validate the proposed method. The simulation of DLOs is based on Obi [30], a unified particle physics engine for deformable objects in Unity3D [31], as shown in Fig. 4. In the simulation, the two ends of the DLO are grasped by two grippers, which can translate and rotate. Both 2D and 3D tasks are tested. In the 2D tasks, the environment dimension  $l$  is 2 and the control input dimension  $n$  is 6; in the 3D tasks  $l=3, n=12$ . In the real-world experiments, the DLOs are placed on a table. One end of the DLO is grasped by a UR5 arm, and the other is fixed. Thus,  $l=2$  and  $n=3$ . The shape of the DLO is represented by 8 features ( $m=8$ ), and the positions of the features are obtained by measuring the markers on the DLO with a calibrated RGB camera in the experiments. Both the data collection frequency and control frequency are 10 Hz.

We choose three representative classes of methods for comparison. The first class is learning forward kinematics models of DLOs offline and using MPC for shape control (FKM+MPC). According to [12], we choose bi-directional LSTM (biLSTM) for modeling and Model Predictive Path Integral Control (MPPI) for control. The second class is estimating the Jacobian matrix online using weighted least square estimation (WLS). We specifically use the method in [19]. The third is based on reinforcement learning. We train an agent using Soft Actor Critic (SAC) [32].

##### A. Offline Learning of the Deformation Model

The offline data of DLOs are collected in simulation, by randomly moving the ends of the DLOs. A RBFN with 256 neurons in the middle layer ( $q = 256$ ) is first trained offline to learn the initial deformation model.

First, we test the offline modeling accuracy on a certain DLO and its relationship to the amount of training data, in which we compare our local linear Jacobian model with nonlinear forward kinematics models based on multi-layer perceptrons (MLP) or biLSTM. Training data and 10k test data are collected on the same DLO. For testing, we use trained models to predict the shape of the DLO after 10 steps. The prediction of the shape at the next step using our method is calculated as  $\hat{\mathbf{x}}_{[t+1]} = \mathbf{x}_{[t]} + \Delta t \hat{\mathbf{J}}(\phi_{[t]}) \dot{\mathbf{r}}_{[t]}$ , where the subscript  $[t]$  represents the variables at step  $t$  and  $\Delta t$  is the step interval. Shown in Fig. 5, the results indicate that our Jacobian model can achieve higher prediction accuracy with less training data. The biLSTM-based model incorporates the physics priors of chain-like DLOs [12], so it performs

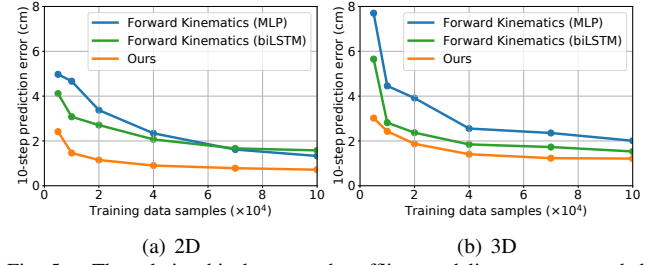


Fig. 5. The relationship between the offline modeling accuracy and the amount of training data. Training data samples and 10k test samples are collected on the same DLO. The error is the average Euclidean distance between the prediction and ground truth of the shape after 10 steps.

TABLE I

COMPARISON OF THE ABSOLUTE STATE INPUT AND RELATIVE STATE INPUT IN OUR METHOD.

Feature velocity prediction relative error <sup>a</sup>	2D		3D	
	New DLO <sup>b</sup>	New DLO +translation <sup>c</sup>	New DLO	New DLO +translation
Absolute state input	30.0%	54.5%	41.1%	81.4%
Relative state input	<b>25.9%</b>	<b>25.9%</b>	<b>36.2%</b>	<b>36.2%</b>

<sup>a</sup> The relative error is calculated as the average of  $\|\hat{\mathbf{x}} - \hat{\mathbf{J}}\dot{\mathbf{r}}\|/\|\hat{\mathbf{x}}\|$ .

<sup>b</sup> A new DLO whose length may be different from the trained DLOs'.

<sup>c</sup> A 0.2m position translation offset is added in each dimension.

better than MLP when the training set is small. Our Jacobian model implies strong local linear prior, which is theoretically and practically reasonable. Hence, the learning efficiency is highly improved.

Second, we further compare the performance of our methods using the absolute state input  $[\mathbf{x}; \mathbf{r}]$  and relative state input  $\phi$  as (6) in the RBFN. We collect data of 10 different DLOs in the simulation. For testing, we perform cross validation, i.e., for each round the model is trained on  $9 \times 3k$  data of 9 DLOs and tested on 10k data of the remaining one. We also test the performance on the test set with constant position translation. The average results of 10 rounds shown in Table I reveal that the relative state input can achieve higher prediction accuracy on new DLOs with different lengths. In addition, when position translation is added, the relative state input is not affected, while the performance of the absolute state input significantly decreases.

##### B. Shape Control with Online Learning

We evaluate the proposed method in DLO shape control tasks and compare it with other methods. All  $m$  features are set as target points for shape control. During the tasks, the offline-trained models are used, and in our method the model will be further updated concurrently. In addition, domain randomization is applied during offline training to improve the models' generalization abilities. All the offline methods are trained on data collected on 10 DLOs with different lengths or diameters in simulation, while our method uses much less data than other offline methods.

Several criteria are defined to evaluate the performance: (1) final task error: the Euclidean distance between the desired shape and final shape within 30s; (2) success rate: if the final task error is less than 5cm, this case is regarded successful; (3) average task error: the average of the final task errors over

TABLE II  
PERFORMANCE OF THE METHODS IN 2D AND 3D DLO SHAPE CONTROL TASKS IN SIMULATION.

Methods	2D tasks				3D tasks			
	Offline training samples	Success rate	Average task error (cm)	Average task time (s)	Offline training samples	Success rate	Average task error (cm)	Average task time (s)
FKM+MPC	180k	82/100	1.662	10.488	180k	52/100	3.298	14.275
WLS	-	85/100	0.992	14.351	-	55/100	1.940	20.214
SAC	1000k	42/100	3.185	6.486	1000k	10/100	3.412	8.070
Ours(w/o online)	<b>30k</b>	94/100	0.461	8.568	<b>30k</b>	69/100	1.446	9.521
Ours	<b>30k</b>	<b>97/100</b>	<b>0.457</b>	8.512	<b>30k</b>	<b>92/100</b>	<b>1.254</b>	9.529

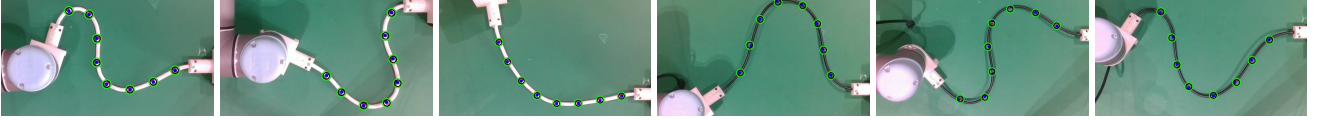


Fig. 6. Some of the shape control tasks accomplished using our method in real-world experiments. The left end of the DLO is grasped by a UR5 arm and the right end is fixed. The green+black circles represent the desired positions of the DLO features. In all cases, the DLO starts from a straight line.

TABLE III  
PERFORMANCE IN REAL-WORLD 2D SHAPE CONTROL TASKS.

Methods	Success rate	Average task error (cm)	Average task time (s)
FKM+MPC	<b>10/10</b>	1.394	7.670
WLS	9/10	1.164	19.089
SAC	1/10	4.955	12.300
Ours(w/o online)	<b>10/10</b>	1.153	10.090
Ours	<b>10/10</b>	<b>0.620</b>	8.220

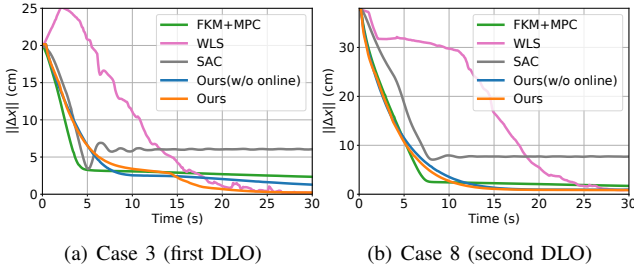


Fig. 7. The control processes in real-world experiments. The  $\|\Delta x\|$  refers to the Euclidean distance between the current shape and the desired shape.

all successful cases; (4) average task time: the average time used to achieve success over all successful cases. Note that the task time is for reference only, since it depends on the control gain in servo methods or the sample range of control input in MPC or RL.

1) *Simulation*: First, we test their performance in both 2D and 3D DLO shape control tasks in simulation. The manipulated DLO is an untrained DLO, and 100 cases with different feasible desired shapes are tested. We also do the ablation study to validate the effect of the online learning in our method. The parameters are set as  $\alpha = 0.3, \eta_1 = 10^{-3}, \eta_2 = 50$ . As shown in Table II, our method significantly outperforms the compared methods on both success rate and average task error, even using much less offline training data. The task time of the online method WLS is the highest because it needs to initialize the Jacobian by moving the DLO ends in each DoF every time it starts. The average task error of FKM+MPC is higher because it has no further updating for the untrained manipulated DLO. The

poor performance of SAC may be due to insufficient training. The contrast is starker in more challenging 3D tasks, where the success rates of compared methods are very low ( $\leq 55\%$ ), including our method without online learning (69%), while our method with online learning achieves a 92% success rate and the lowest average task error.

2) *Real-world experiments*: We also evaluate these methods in real-world 2D tasks. The same offline models as those in the simulation are used, which means no real-world data are collected for offline training. We separately carry out 5 tests with different feasible desired shapes on two DLOs: an electric wire with a length of 0.45m and diameter of 8mm, and an HDMI cable with a length of 0.6m and diameter of 5mm, as shown in Fig. 6. The parameters are set as  $\alpha = 0.3, \eta_1 = 10^{-3}, \eta_2 = 200$ . The results are shown in Table III. Fig. 7 visualizes the control processes of two cases. Since the control input dimension is only 3, all methods perform well in these relatively simple tasks except SAC. It is shown that the processes of WLS are slow and unsmooth, and those of FKM+MPC are fast but less precise. Our method completes all 10 tasks and achieves the lowest average task error, where the online learning enables faster and more precise control.

## V. CONCLUSION

This paper considers the shape control of DLOs with unknown deformation models. We formulate the deformation model in a local linear format, which is estimated in both offline and online phases. First, the offline learning well initiates the estimation of the model. Then, the adaptive control scheme with online learning further updates the model and achieves the shape control tasks in the presence of an inaccurate offline model. The experiments demonstrate that the offline learning of the local linear deformation model is accurate and data-efficient. By combining the offline and online learning, our method outperforms the compared methods, and adapts well to untrained DLOs. Future work will include more detailed analysis of our method and validations in real-world 3D dual-arm manipulation tasks using a high-precision 3D camera.

## REFERENCES

- [1] X. Li, X. Su, Y. Gao, and Y. Liu, "Vision-based robotic grasping and manipulation of usb wires," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3482–3487.
- [2] S. Jin, D. Romeres, A. Ragnathan, D. K. Jha, and M. Tomizuka, "Trajectory optimization for manipulation of deformable objects: Assembly of belt drive units," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [3] L. Cao, X. Li, P. T. Phan, A. M. H. Tiong, H. L. Kaan, J. Liu, W. Lai, Y. Huang, H. M. Le, M. Miyasaka, K. Y. Ho, P. W. Y. Chiu, and S. J. Phee, "Sewing up the wounds: A robotic suturing system for flexible endoscopy," *IEEE Robotics Automation Magazine*, vol. 27, no. 3, pp. 45–54, 2020.
- [4] R. Laezza, R. Gieselmann, F. T. Pokorný, and Y. Karayiannidis, "Reform: A robot learning sandbox for deformable linear object manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [5] W. Wang, D. Berenson, and D. Balkcom, "An online method for tight-tolerance insertion tasks for string and rope," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2488–2495.
- [6] H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/un knotting manipulation of deformable linear objects," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 371–395, 2006.
- [7] D. McConachie, T. Power, P. Mitran, and D. Berenson, "Learning when to trust a dynamics model for planning in reduced state spaces," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3540–3547, 2020.
- [8] C. Jiang, A. Nazir, G. Abbasnejad, and J. Seo, "Dynamic flex-and-flip manipulation of deformable linear objects," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3158–3163.
- [9] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, 2021.
- [10] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," in *4th Conference on Robot Learning (CoRL)*, 2020.
- [11] W. Zhang, K. Schmeckpeper, P. Chaudhari, and K. Daniilidis, "Deformable linear object prediction using locally linear latent dynamics," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [12] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2372–2379, 2020.
- [13] Y. Yang, J. A. Stork, and T. Stoyanov, "Learning to propagate interaction effects for modeling deformable linear objects dynamics," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [14] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *4th Conference on Robot Learning (CoRL)*, 2020.
- [15] L. Rita and K. Yiannis, "Learning shape control of elastoplastic deformable linear objects," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [16] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y. Liu, F. Zhong, T. Zhang, and P. Li, "Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 429–441, 2016.
- [17] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 479–484.
- [18] S. Jin, C. Wang, and M. Tomizuka, "Robust deformation model approximation for robotic cable manipulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6586–6593.
- [19] R. Lagneau, A. Krupa, and M. Marchal, "Automatic shape control of deformable wires based on model-free visual servoing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5252–5259, 2020.
- [20] S. Duenser, J. M. Bern, R. Poranne, and S. Coros, "Interactive robotic manipulation of elastic objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3476–3481.
- [21] A. Koessler, N. Roca Filella, B. Bouzgarrou, L. Lequievre, and J.-A. Corrales Ramon, "An efficient approach to closed-loop shape control of deformable objects using finite element models," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [22] M. Rambow, T. Schaub, M. Buss, and S. Hirche, "Autonomous manipulation of deformable objects based on teleoperated demonstrations," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2809–2814.
- [23] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2146–2153.
- [24] T. Tang, C. Wang, and M. Tomizuka, "A framework for manipulating deformable linear objects by coherent point drift," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3426–3433, 2018.
- [25] T. Bretl and Z. McCarthy, "Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 48–68, 2014.
- [26] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [28] H. Yu, T. Xie, S. Paszczynski, and B. M. Wilamowski, "Advantages of radial basis function networks for dynamic system design," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 12, pp. 5438–5450, 2011.
- [29] S. Arimoto, *Control theory of nonlinear mechanical systems*. Oxford University Press, 1996.
- [30] V. M. Studio. (2019) Obi - Unified particle physics for Unity 3D. [Online]. Available: <http://obi.virtualmethodstudio.com/>
- [31] U. Technologies. (2021) Unity real-time development platform. [Online]. Available: <https://unity.com/>
- [32] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.