POLITECNICO DI TORINO
Repository ISTITUZIONALE

Deformable Linear Objects Manipulation With Online Model Parameters Estimation

Original
Deformable Linear Objects Manipulation With Online Model Parameters Estimation / Caporali, Alessio; Kicki, Piotr; Galassi, Kevin; Zanella, Riccardo; Walas, Krzysztof; Palli, Gianluca. - In: IEEE ROBOTICS AND AUTOMATION LETTERS. - ISSN 2377-3766. - 9:3(2024), pp. 2598-2605. [10.1109/LRA.2024.3357310]

Availability:
This version is available at: 11583/2991570 since: 2024-08-06T15:50:00Z

Publisher:
IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS

Published
DOI:10.1109/LRA.2024.3357310

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

03 June 2026

Deformable Linear Objects Manipulation With Online Model Parameters Estimation

Alessio Caporali, Piotr Kicki, *Member, IEEE*, Kevin Galassi, Riccardo Zanella, Krzysztof Walas, *Member, IEEE*, and Gianluca Palli, *Senior Member, IEEE*

***Abstract*—Manipulating deformable linear objects (DLOs) is a challenging task for a robotic system due to their unpredictable configuration, high-dimensional state space and complex nonlinear dynamics. This letter presents a framework addressing the manipulation of DLOs, specifically targeting the model-based shape control task with the simultaneous online gradient-based estimation of model parameters. In the proposed framework, a neural network is trained to mimic the DLO dynamics using the data generated with an analytical DLO model for a broad spectrum of its parameters. The neural network-based DLO model is conditioned on these parameters and employed in an online phase to perform the shape control task by estimating the optimal manipulative action through a gradient-based procedure. In parallel, gradient-based optimization is used to adapt the DLO model parameters to make the neural network-based model better capture the dynamics of the real-world DLO being manipulated and match the observed deformations. To assess its effectiveness, the framework is tested across a variety of DLOs, surfaces, and target shapes in a series of experiments. The results of these experiments demonstrate the validity and efficiency of the proposed methodology compared to existing methods.**

***Index Terms*—Deformable linear objects, manipulation, shape control.**

I. Introduction

ROBOTIC solutions involving Deformable Linear Objects (DLOs) like ropes, cables, hoses, and wiring harnesses are highly complex, presenting various challenges from two main perspectives: perception [1] and manipulation [2]. From a perception standpoint, dealing with DLOs is a tough task. Their ambiguity can make it difficult to distinguish different parts of them or to distinguish DLOs from other objects [3], especially given their relatively small size [4], [5]. Moreover, the detection of the full DLO state, including the twist along the DLO, poses also a challenge [6], [7]. Regarding manipulation, DLOs prove to be challenging due to their unpredictable configuration behavior, high dimensional state-space, and complex nonlinear dynamics [2], [7]. Therefore, a deep understanding of their physical characteristics is essential for predicting and controlling their shape effectively [8].

In this letter, a manipulation framework exploiting a physical prior of DLOs dynamics is proposed. In particular, a data-driven learned model of the DLOs' dynamics is developed to predict the DLO behavior under manipulative actions. The prediction is made using a Neural Network (NN) trained to approximate the dynamics of a class of DLOs, by conditioning its predictions on the set of the analytical model parameters.

First, the DLO's dynamics is modeled by a set of differential equations describing the DLO as point masses connected by axial and torsional springs [9], obtaining a so-called analytical DLO model. Then, the analytical DLO model is used to generate a comprehensive dataset by systematically sampling a variety of model parameters, diverse DLO configurations, and various manipulation actions. Consequently, a neural network is trained utilizing this generated dataset, as visualized in the training phase depicted in Fig. 1. Notably, the neural network is conditioned over the model parameters, such that it can be easily adapted to match different real-world DLOs.

The obtained neural network model is employed during the online phase in Fig. 1 to estimate the manipulation actions to steer the DLO from its initial to a final target configuration, performing a task commonly referred to as shape control. In this context, the adoption of the network model is preferred over the

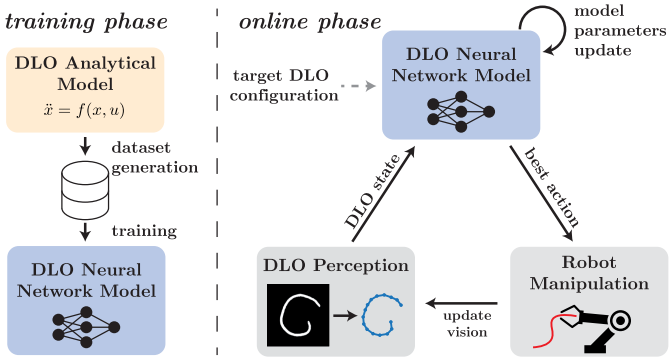


Fig. 1. Schematic view of the two phases composing the proposed manipulation framework: 1) training phase for dataset generation and NN training and 2) online phase for simultaneous estimation of the best action and model parameters during the shape control task.

Manuscript received 26 September 2023; accepted 16 January 2024. Date of publication 23 January 2024; date of current version 2 February 2024. This letter was recommended for publication by Associate Editor Michael C. Welle and Editor Júlia Borràs Sol upon evaluation of the reviewers' comments. This work was supported by the European Union's Horizon 2020 research and innovation programme under Grant 870133 through RIA Project REMODEL (Robotic technologies for the manipulation of complex deformable linear objects). *(Corresponding author: Alessio Caporali.)*

Alessio Caporali, Kevin Galassi, Riccardo Zanella, and Gianluca Palli are with the DEI - Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: alessio.caporali2@unibo.it; kevin.galassi2@unibo.it; r.zanella@unibo.it; gianluca.palli@unibo.it).

Piotr Kicki and Krzysztof Walas are with the Institute of Robotics and Machine Intelligence, Poznan University of Technology, 60-965 Poznan, Poland (e-mail: piotr.kicki@put.poznan.pl; krzysztof.walas@put.poznan.pl).

Project website at https://sites.google.com/view/dlo-manipulation.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3357310, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3357310

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

analytical model due to its computation efficiency, stability, and scalability.

The proposed method uses a gradient-based approach to estimate the best manipulation action to achieve a desired target configuration, by minimizing the error between the network prediction and the desired DLO configuration. A similar gradient-based approach is exploited to estimate the model parameters of the DLO, where the estimation is performed by minimizing the error between the model prediction and the observed real-world DLO configuration obtained after a manipulation action.

The proposed framework can directly be applied for the manipulation of various DLOs on diverse surfaces, thanks to the data-driven approximation of the DLO dynamics conditioned on the model parameters. Therefore, there is no need to: 1) generate every time new task-specific data as in [7], 2) introduce complex online adaptation controllers as in [2], [10], 3) perform cumbersome and not intuitive parameters identification procedures as in [8], [11].

In summary, the contributions of this letter are:

- NN-based DLO dynamics approximation conditioned on several analytical model parameters allowing adaptability to different real-world scenarios;
- Efficient gradient-based action prediction and parameters estimation employing the same learned NN model;
- Experimental validation of the method on different real-world DLOs and surfaces.

In the following, the related works are discussed in Section II. The proposed framework is detailed in Section III. The experimental evaluation of the method is provided in Section IV and the conclusions are drawn in Section V.

II. Related Works

A. DLO Shape Control Task

The term shape control of DLOs is typically used to refer to two different manipulation problems targeting the achievement of a final target shape: 1) the manipulation of a *soft* DLO with a sequence of pick and place actions [12], [13], [14], where the deformation of the DLO is held in place by the friction of the surface underneath. 2) the manipulation of *elastic* DLOs with one or more robotic arms and/or one end of the DLO fixed [2], [7], [10], [15], [16], where the second arm is used to achieve better control of the shape, e.g. in the situation where the DLO's stiffness and the object exhibit rigid or plastic behavior. The outcome of the task can be assessed by comparing the achieved configuration to the target one in two possible ways [17]: by measuring their *relative similarity*; by evaluating their *absolute similarity* (i.e. a more challenging alternative considering also the final positioning in space). In this letter, the latter is employed.

B. Model-Free Approaches

One of the popular approaches to manipulating DLOs is to use methods that do not require nor create DLO models. Examples of these methods are those based on expert demonstration. In [18], shape similarity is used to determine which human demonstration should be replayed by the robot to achieve the goal. Whereas in [19], human expertise was used to learn the DLO manipulation policy. This kind of policy can be also learned without supervision in a reinforcement learning paradigm, however, it is typically done only in simulation [17], which limits the potential application to the real DLOs. To approach this *reality-gap* and improve the robustness, authors of [20], [21], [22] focused on the online adaptation of the DLO control strategy. In the case of [22], the parameters of the controller were adjusted online based on the tracking error. Whereas in [20], [21], the control law relies on the Jacobian that locally approximates how the movement of the grippers affects the DLO. These methods, which utilize local approximations of DLO motions and online adaptation of controller parameters, have the potential to enhance the system's manipulation abilities in the context of model-free approaches. Nevertheless, model-based approaches can typically strengthen the system's robustness through its better generalizability to diverse scenarios.

C. Learned DLO Models

The literature related to learning-based methods can be divided by the type of DLO to be manipulated. Concerning the manipulation of *soft* DLOs like ropes with pick and place actions, in [13] an image-based predictive DLO model is learned in a self-supervised manner. Instead, in [14], the image of the DLO is embedded in the latent space with linear dynamics imposed on it. Differently, [12] proposes learning the DLO dynamics in state-space while enforcing the physic priors via a biLSTM architecture modeling the DLO as a chain-like mass-spring system. In all these works, manipulation actions are sampled randomly and the best one is selected considering suitable cost functions. On the contrary, a gradient-based approach for estimating the best action is proposed in this work, where also a rotation component is estimated. In this way, a more complex manipulation action can be executed with respect to the simpler linear displacement operation. Regarding the manipulation of *elastic* DLOs, several works proposed a learning-based framework to predict the DLO dynamics. In [2], [7], [10] the DLO dynamic is approximated with data-driven approaches trained using simulation data. In [2] the authors propose an online adaptation of the DLO model to compensate for the *reality gap*. Similarly, in [10] a linear residual model is estimated online.

D. DLO Analytical Models and Physical Parameters

Many different physical models of DLOs have been proposed in the past [8], ranging from simpler mass-spring [9] and energy-based models to more accurate but computationally demanding elastic-rod, dynamic splines, and finite element models [8]. Other than the selection of a specific model, the choice of the integration method is crucial to achieve a good trade-off between simulation accuracy and efficiency, and different integration approaches like Runge-Kutta and symplectic have been proposed [23]. As opposed to the mentioned force-based methods, a differentiable position-based simulation of DLOs is proposed in [11] where the integration steps are avoided leading to a more efficient and stable simulation.

Despite the many models available, their application in robotic systems is usually marginal due to the high computational cost and sensitivity to the choice of physical and simulation parameters. Indeed, their estimation is a cumbersome task, as can be seen in [8] where the physical model parameters are constantly adjusted to make the simulation results approach the experimental ones. Alternatively, in [11], the differentiability of the framework is exploited for the estimation of model parameters.

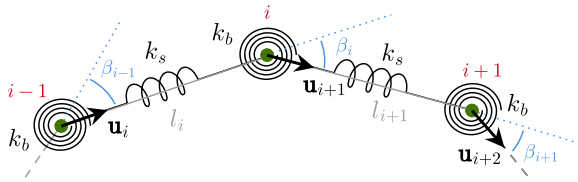


Fig. 2. DLO analytical model representation.

However, the process is quite slow and can not be performed in an online framework.

Learning-based methods usually employ a DLO simulator to collect training data, where the simulator is generally constructed based on one of the mentioned DLO models. However, only a few learning-based works pay attention to DLO parameters. In [14] and [7] the DLO parameters employed in the simulator are estimated by optimizing the simulation against a small set of real samples, employing a sampling-based method [14] or a differential evolution strategy [7].

Compared to the parameters estimation procedures of [7], [8], [11], [14], this letter proposes an efficient gradient-based procedure which can seamlessly be performed online during the execution of the shape control task.

III. Method

The proposed framework focuses on shaping DLO toward a desired target. Simultaneously, during task execution, the framework also estimates the model parameters to enhance its ability to capture the dynamics of the actual manipulated DLO. The framework exploits two different phases (see Fig. 1), namely a training phase and an online phase. In the first one, the neural network-based DLO model is trained to mimic the behavior of the analytical DLO model for a wide range of its parameters. While in the online phase, the trained NN model is used to predict actions and simultaneously optimize the parameters so that the NN model predictions, which are conditioned on these parameters, match the observed behavior of the DLO.

The DLO analytical model is introduced in Section III-A, and the DLO state representation is detailed in Section III-B. The NN model is presented in Section III-C whereas the gradient-based estimation of the action and model parameters is discussed in Section III-D. Finally, the shape control task with online parameters adaptation is presented in Section III-E.

A. Analytical Model

A DLO can be physically modeled via a set of nodes having proper mass and axial springs connecting the nodes to create a serial chain [9], as shown in Fig. 2. In addition, the bending stiffness of the DLO is modeled by placing a torsional spring at each node. To improve the stability of the model, a viscous friction proportional to the velocity of the node is included as a damping term.

The dynamics of the generic node i can be written as:

$$m_i \ddot{\mathbf{p}}_i = -k_d \dot{\mathbf{p}}_i + \mathbf{f}_i^s + \mathbf{f}_i^b, \tag{1}$$

where $\mathbf{p}$ is the node coordinates, k_d a damping constant, f_i^s the force due to the axial effects and f_i^b the forces due to the bending effects. The axial effects f_i^s are computed as:

$$\mathbf{f}_i^s = -k_s(l_i - l_i^0)\mathbf{u}_i + k_s(l_{i+1} - l_{i+1}^0)\mathbf{u}_{i+1}, \tag{2}$$

where l_i and l_i^0 are the current and initial lengths of link i respectively, while u_i represent the unit vector of node i. With reference to Fig. 2, the bending effect can be written as:

$$\begin{aligned}\mathbf{f}_i^b = {} & k_b \frac{\beta_{i-1}}{l_i \sin\beta_{i-1}} \mathbf{u}_i \times (\mathbf{u}_{i-1} \times \mathbf{u}_i) \\ & - k_b \frac{\beta_i}{l_i \sin\beta_i} \mathbf{u}_i \times (\mathbf{u}_i \times \mathbf{u}_{i+1}) \\ & - k_b \frac{\beta_i}{l_{i+1} \sin\beta_i} \mathbf{u}_{i+1} \times (\mathbf{u}_i \times \mathbf{u}_{i+1}) \\ & + k_b \frac{\beta_{i+1}}{l_{i+1} \sin\beta_{i+1}} \mathbf{u}_{i+1} \times (\mathbf{u}_{i+1} \times \mathbf{u}_{i+2}),\end{aligned} \tag{3}$$

where

$$\beta_i = \arctan \frac{||\mathbf{u}_{i+1} \times \mathbf{u}_i||}{\langle \mathbf{u}_{i+1}, \mathbf{u}_i \rangle}$$

represents the angle between u_i and u_{i+1}.

The manipulation action executed on the DLO model is parametrized as a pick-and-place operation executed on the edge of the DLO, i.e. between two consecutive nodes. The decision to perform actions at the edge level is primarily influenced by the physical design of the gripper. In fact, the gripper does not interact with the DLO at a single node, but it engages with the DLO in a manner that can be more accurately described as the movement of the segment between two consecutive nodes. The DLO action parameters vector is defined by

$$a = [\alpha, \delta_x, \delta_y, \delta_\theta], \tag{4}$$

where α denotes the index of the edge to grasp, δ_x and δ_y are the linear displacements applied to the selected edge $\{\mathbf{p}_\alpha, \mathbf{p}_{\alpha+1}\}$ and δ_θ is the rotation applied to the initial edge orientation. The effect of the action introduced above is simulated using forward Euler method applied to the discretized version of (1).

B. DLO Perception and State Representation

Since the DLO's dynamics is based on a mass-spring-damper system, an appropriate representation according to the chosen model is utilized. The DLO state V is represented as a sequence of n 2D points in the Cartesian space, i.e. $V \in \mathbb{R}^{n\times 2}$. In the simulation, each node represents the position of the simulated masses. The 2D coordinates of the DLO in the real scenario are obtained using *RT-DLO* [1], an algorithm for real-time DLO perception, where the input image is provided by a fixed 3D vision sensor. From the acquired point cloud, the workspace plane is segmented out to obtain the DLOs points in the scene. These points are then projected on the image plane by utilizing the camera's intrinsic parameters obtaining a binary mask of the DLO in the scene. The binary mask is forwarded to *RT-DLO* that performs the modeling of the DLO as a line graph representation of the DLO, i.e. a sequence of nodes and edges, as shown in Fig. 1. Therefore, the node coordinates in the line graph represent the state V.

C. Neural Network Model

A NN is used to approximate the analytical DLO model, significantly improving computational efficiency. Indeed, the complexity of the analytical DLO model (1)–(3) affects its

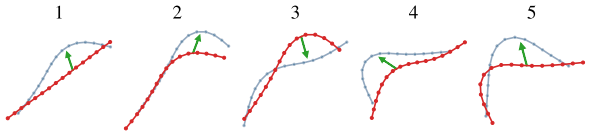


Fig. 3. Sequence of $k = 5$ dataset samples generated with the simulated DLO. V_{in} in red, V_{out} in blue and action in green.

performance and makes using it in an online framework challenging. Instead, a NN can be trained to accurately replicate the DLO dynamics by exploiting a dataset of DLO movements, which can be generated offline using the analytical DLO model. Thanks to the use of a relatively small neural network, a constant and short inference time is obtained which is more than an order of magnitude smaller than the time needed to evaluate the analytical DLO model. While the analytical model can be hard to differentiate, the NN model is easily differentiable wrt the parameters, improving the possibility of optimizing all relevant tasks.

1) Dataset Generation: The dataset is generated by simulating the analytical DLO model subjected to a set of random actions. Each data sample consists of the DLO initial and final configurations, the performed action, and the employed model parameters. The DLO initial and final configurations are sets of 2D points characterizing the DLO state, i.e. V_{in} and V_{out}, while the action is described by parameters introduced in (4).

The DLO axial deformation is assumed to be negligible for the purposes of this work, thus the coefficient k_s is kept fixed to a high value. Instead, the damping term k_d, the bending term k_b, the length of the DLO, and the mass of the DLO change within predefined ranges. In particular, both the length and the mass are assumed to be known quantities since they can usually be measured. Eventually, the DLO length can be estimated online using the perception algorithm and the mass can be measured using force sensors mounted on the robot. The other two terms are instead difficult to measure, so they are estimated online.

Aiming to learn a general DLO model, both the action and model parameters are drawn from a broad range of values covering the expected real-world variability. In particular, each value is sampled from a uniform distribution with specific boundaries, except for the edge index which is sampled from a discrete uniform distribution.

To generate the dataset, the physical parameters are set to random values from the physically plausible ranges, and the simulated DLO is initialized with an almost linear initial configuration. Then, a set of k actions is sampled and the behavior of DLO is simulated after applying them sequentially. This procedure is exploited in order to build a diversified dataset in terms of DLO configuration complexity. In Fig. 3 an example sequence is shown. After the execution of each action, a dataset sample is saved containing the reached DLO configuration, the initial DLO configuration, the model parameters, and the performed action.

2) Data Augmentation: To improve the training efficiency and generalization capabilities of the NN model, several augmentation and normalization strategies are implemented on the data. The idea is to exploit the symmetries in the DLO data to reduce the amount of information the NN has to learn.

The normalization is performed by finding a transformation that makes V_{in} aligned to the x-axis and mean-centered, and applying it to normalize both V_{in} and V_{out}. This transformation

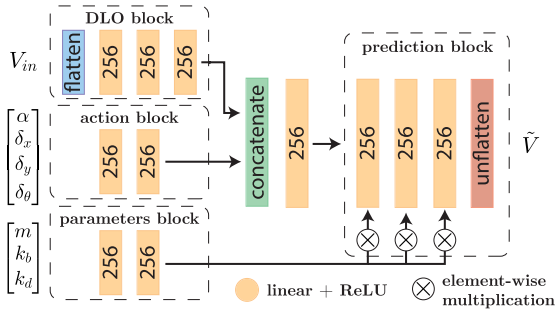


Fig. 4. Neural network architecture.

is composed of the translation equal to the negative geometrical center of the V_{in} and rotation that is required to make the first principal component of the points V_{in} aligned to the x-axis. The nodes are ordered from negative to positive x values by flipping V_{in} and V_{out} along the rows if necessary. The action index α is adjusted accordingly. In addition, the action parameters are scaled to be within the [0,1] range for α and $[-1, 1]$ range for the displacements. The model parameters are also normalized within the [0,1] range.

3) Neural Network Architecture: The neural network architecture is based on a set of Linear layers followed by ReLU activation functions. In detail, the network is composed of four main blocks illustrated in Fig. 4: the action block, the physical parameters block, the DLO block, and the prediction block. The input of the network is the initial configuration of the DLO V_{in}, the action parameters a, and the model parameters p. The length is not provided as input since it is implicit from V_{in}, thus p denotes only the model parameters provided as input, i.e. $p = [m, k_b, k_d]$. The output of the network, denoted as $\tilde{V}$, is the sequence of predicted changes of the 2D DLO coordinates from the initial configuration. The final predicted DLO configuration V_{pred} is expressed as

$$V_{\text{pred}} = \mathcal{F}(V_{\text{in}}, a, p) = \tilde{V}(V_{\text{in}}, a, p) + V_{\text{in}}. \tag{5}$$

The network is trained to minimize the mean squared error between the predicted V_{pred} and the expected V_{out} final configurations.

D. Gradient-Based Estimation of Action and Parameters

The trained NN model is used to estimate both the next manipulation action and the parameters that allows for accurate approximation of the observed DLO behavior. These two estimation procedures are performed using numerical optimization of the loss function between two DLO states. This loss is computed as the sum of L2 norms between corresponding points among the two states and can be defined by

$$\mathcal{D}(V_1, V_2) = \sum_{i=1}^{n} \|V_{1,i} - V_{2,i}\|. \tag{6}$$

An illustration of this idea is provided in Fig. 5. Since the NN model is intrinsically differentiable, a gradient-based approach can be used for the optimization of the above-mentioned loss function. In addition, thanks to the possibility of performing the optimization in batch, the mentioned gradient-based optimization can be performed quite efficiently.

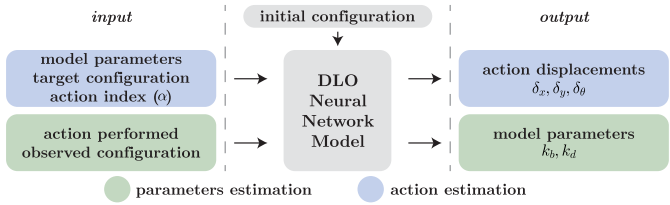


Fig. 5. Gradient-based action and DLO parameters estimation.

1) Action Estimation: For the best action estimation given the current DLO state V_{in} and the model parameters p, the action parameters a minimizing the difference between the NN prediction $\mathcal{F}(V_{\text{in}}, a, p)$ and the target shape V_{tgt}, i.e. the goal to be reached in the shape control task, are sought. While this can be easily done using gradients for the edge displacement parameters, this is not the case for the edge index. Thus, the efficient batch processing capabilities of the NN model are employed, and $n-1$ optimizations are executed simultaneously, with the assumption that the edge index is held constant in each of them. Finally, the best action among the ones evaluated for each edge is selected. This optimization procedure can be in general described by

$$a^* = \underset{a}{\text{argmin}}\ \mathcal{D}(\mathcal{F}(V_{\text{in}}, a, p), V_{\text{tgt}}), \tag{7}$$

where a^* denotes the action that maximally reduces the difference between V_{in} and V_{tgt} according to the used NN model $\mathcal{F}$. However, as mentioned, in practice first are performed $n-1$ optimizations of the form

$$a_j^* = [j, \underset{\delta_x,\delta_y,\delta_\theta}{\text{argmin}}\ \mathcal{D}(\mathcal{F}(V_{\text{in}}, [j, \delta_x, \delta_y, \delta_\theta], p), V_{\text{tgt}})], \tag{8}$$

where a_j^* denotes the best action obtained for the fixed edge index j, and then the best action a^* is sought from the a_j^* by

$$a^* = \underset{a_j^* \text{ for } j\in\{1,2,\ldots,n-1\}}{\text{argmin}}\ \mathcal{D}(\mathcal{F}(V_{\text{in}}, a_j^*, p), V_{\text{tgt}}). \tag{9}$$

2) Parameters Estimation: Similarly to actions, the model parameters are estimated by searching for the ones that minimize the difference between the NN prediction V_{pred} and the observed DLO state V_{out}. This optimization can be written by

$$k_b^*, k_d^* = \underset{k_b,k_d}{\text{argmin}}\ \mathcal{D}(\mathcal{F}(V_{\text{in}}, a, p), V_{\text{out}}) \tag{10}$$

where k_b^*, k_d^* denotes the optimal values of the bending and damping coefficients, and $p = [m, k_b, k_d]$. Since the mass m can be measured, it is not optimized but measured and provided as input to the NN model.

E. Shape Control Task With Online Parameters Adaptation

To improve the manipulation capabilities of the robotic system in the case of the shape control task for a real previously unseen DLOs, an approach that utilizes both model-based DLO shape control and the online model parameters adaptation is proposed. This can be achieved by jointly using the gradient-based action and parameters optimization routines developed in Section II-I-D. In Algorithm 1, the proposed method is detailed. If there is no prior knowledge, the model parameters k_d and k_b can be initialized at the midpoint of the range used in the dataset generation, see Section III-C1. Therefore, given a target shape

Algorithm 1. Online Adaptation.

Input: V_{tgt}
Output: V_{out}, k_d, k_b

1 $k_d, k_b \leftarrow k_{d,\text{start}}, k_{b,\text{start}}$
2 $\mathcal{V}_{in}, \mathcal{V}_{out}, \mathcal{A}^* \leftarrow \emptyset$
3 $V_{\text{in}} \leftarrow$ get_dlo_state()
4 $\epsilon \leftarrow \mathcal{D}(V_{\text{in}}, V_{\text{tgt}})/n$
5 **while** $\epsilon > \epsilon_{th}$ **do**
6 $\quad a^* \leftarrow$ best_action$(V_{\text{in}}, V_{\text{tgt}}, k_d, k_b)$
7 $\quad$ robot_manipulation(a^*)
8 $\quad V_{\text{out}} \leftarrow$ get_dlo_state()
9 $\quad \mathcal{V}_{\text{in}}, \mathcal{V}_{\text{out}}, \mathcal{A}^* \leftarrow$ update_dataset$(V_{\text{in}}, V_{\text{out}}, a^*)$
10 $\quad k_d, k_b \leftarrow$ best_parameters$(\mathcal{V}_{\text{in}}, \mathcal{V}_{\text{out}}, \mathcal{A}^*)$
11 $\quad \epsilon \leftarrow \mathcal{D}(V_{\text{out}}, V_{\text{tgt}})/n$
12 $\quad V_{\text{in}} \leftarrow V_{\text{out}}$

V_{tgt}, the robotic system iterates (line 5) executing a sequence of manipulation actions until the error between the current observed state V_{out} and the target shape V_{tgt} computed according to (6) is below a user-defined threshold ϵ_{th}.

At each iteration, the camera system is first used to capture a new sample from the scene and process it via the perception system described in Section III-B, thus obtaining an initial configuration (line 3). Then, the best action to move the DLO toward the target configuration is computed (line 6) according to (7), and the result of the performed action on the real system is observed (line 8). The interaction with the real system is saved (line 9) into a task dataset, which is initialized empty at the beginning of the task. The task dataset is used for the best parameters estimation performed following (10) (line 10). Finally, the configuration error is updated (line 11) by comparing the achieved shape to the target one.

IV. Experiments

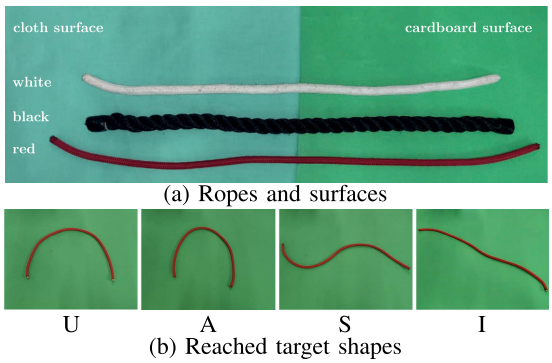
The manipulation framework of Section III is evaluated in the context of a shape control task, with a robotic setup composed a Panda Robot equipped with a parallel-jaw gripper, a Photoneo Motioncam3D statically mounted on the robotic cell, and a selection of ropes and surfaces.

Three ropes are used in the experiments: a *white* rope (0.45m, 0.02kg, 0.01m diameter); a *black* rope (0.42m, 0.05kg, 0.014m diameter); and a *red* rope (0.50m, 0.02kg, 0.005m diameter). Note, the *black* rope is the stiffest one, while the *red* rope exhibits a higher degree of bending elasticity compared to the *white* rope. Additionally, two planar surfaces with different physical properties are used: a *cloth* and a *cardboard* surface. The *cardboard* is smoother and more slippery than the *cloth*.

The experiments were executed employing as hardware an Ubuntu PC equipped with an Intel CPU i7-12700H clocked at 2.3GHz and an Nvidia GPU 3050Ti.

A. Optimization Details

The NN model (Section III-C) is trained on a dataset of DLO manipulation samples, obtained by the analytical model (Section III-A), comprising about 275K elements. The dataset is generated by employing the following boundary values for action and model parameter ranges. Regarding the action, $\alpha \in$



(a) Ropes and surfaces

(b) Reached target shapes

Fig. 6. Experimental robotic setup comprising different ropes and surfaces (a) Target shapes achieved with the *red* rope on the *cardboard* surface (b).

$[0, 15]$; δ_x and δ_y are confined at ± 0.10 m; and δ_θ is within $\pm\pi/4$ rad. Concerning the model parameters, the damping coefficient $k_d \in [3, 30]$ Ns/m; the bending coefficient $k_b \in [0.05, 1.0]$ N/m; the DLO length within $[0.15, 0.50]$ m; the mass within $[0.02, 0.1]$ kg.

1) NN Model Training: A 90-10% split is employed to organize the dataset into training and validation sets. The network is trained for 100 epochs (batch size 128, learning rate 5×10^{-5}). The final weights are selected as the ones having the minimum mean squared error validation loss.

2) Action and Parameters Estimation: The gradient-based action estimation of Section III-D is performed for 500 optimization steps employing a learning rate of 0.005. Regarding the estimation of the parameter, the optimization is performed for 3000 steps with a learning rate of 0.01. In both cases, an early stopping procedure is implemented in case of the earlier convergence. The tanh and sigmoid activation functions are employed to limit the normalized action and model parameters, respectively, to the ranges of $[-1, 1]$ and [0,1] (Section III-C2). This ensures to obtain denormalized values consistent with the boundaries employed in the dataset.

B. Shape Control Task With Online Parameters Estimation

The shape control task involves the manipulation of four distinct target shapes: the *U*, *A*, *S* and *I* shapes (see Fig. 6(b)). Notably, the *A* shape differs from the *U* shape by requiring a more pronounced bending in its central region. Conversely, the *S* shape is characterized by two opposing and symmetric bends. The *I* shape is used to evaluate the situation of zero curvature target, where the *I* target is rotated by 90deg with respect to the initial configuration. The reachability of all these shapes was ensured by rearranging the ropes between the target and initial configurations using a single human arm restricted to the motions available to the robot.

The task is performed following the online adaptation approach introduced in Section III-E. The initial DLO configuration V_{in} is a straight line. The initial model parameters k_d, k_b are selected around the mid values of their ranges in the dataset, i.e. $k_d = 14$ and $k_b = 0.5$. The task is executed for each target shape V_{tgt} on each planar surface 5 times. The execution of the task is terminated once the error computed according to (6) between V_{out} and V_{tgt} is below 0.01m.

The results of the experiments are provided in Fig. 7, where, within each subplot of a specific rope, columns illustrate the task execution for specific target shapes, while rows provide an analysis of error and model parameters. In detail, the first row focuses on the mean error, with a dashed horizontal line denoting the 0.01m threshold marking the completion of the task. The second and third rows delve into the examination of the bending parameter k_b and the damping parameter k_d respectively. Here, the dashed lines represent the estimated model parameters derived from all samples across all repetitions performed for a given shape. These values, in essence, serve as potential reference values for the specific parameters.

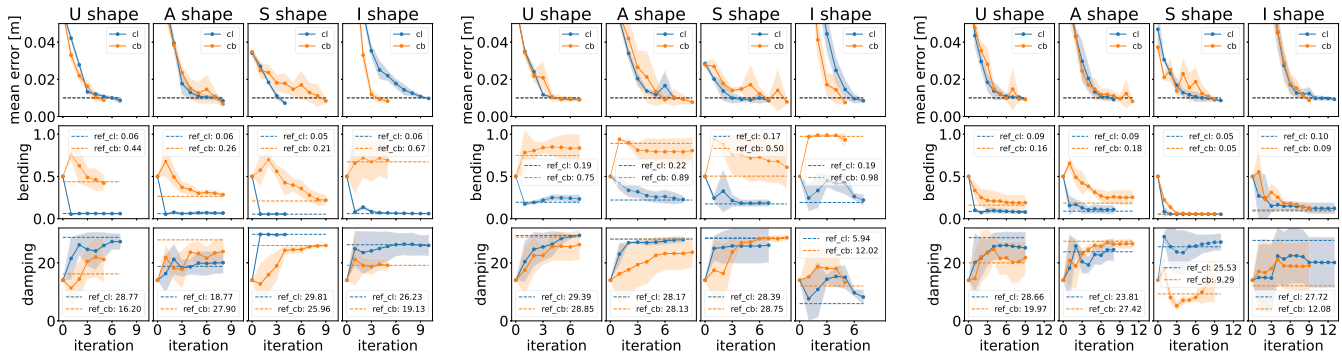
Analyzing the x-axis in the plots, iteration 0 represents the initial condition with a straight DLO configuration and model parameters at their initial values. An action is then executed by the robotic system, updating the observed DLO configuration. Model parameters are recalculated based on a single data sample, resulting in updated values at manipulation iteration 1. This iterative process continues until the specified termination condition is met. At manipulation iteration m, the parameter estimation is based on m data samples.

Examining the plots in Fig. 7, it is worth noting that similar bending parameters are consistently estimated for each specific rope on the *cloth* surface, regardless of the chosen target shape. The parameters estimated on the *cardboard* surface exhibit a higher degree of variability, indicating the presence of more complex dynamics due to increased slippage. The estimation of the damping term is less stable. In general, different pairs of k_d and k_b values are estimated for the same rope on different surfaces, highlighting the adaptation processes. The estimated bending parameters comparison confirms significantly different physical properties between the three ropes and that the *black* rope is the stiffest one, as initially predicted. For instance, on the *cloth* surface, the reference bending values are approximately 0.06 and 0.08 for the *white* and *red* ropes and about 0.19 for the *black* rope.

To gain a deeper insight into the impact of the online model parameters estimation, Fig. 8 presents a comparison among *mid-range*, *online* estimated, and *best* parameters. The latter refers to those estimated at the end of each task repetition, while the *mid-range* to the ones from which *online* estimation starts. These parameter setups were compared using the mean prediction error, denoted as $\mathcal{D}(V_{\text{pred}}, V_{\text{out}})$, computed after each iteration of the shape control task across all the target shapes. The plots illustrate how, within just a few iterations, the proposed method attains parameters that yield a mean error between V_{pred} and V_{out} comparable to the *best* scenario, and in most of the cases significantly better than for the *mid-range* parameters.

C. Comparison With State-of-The-Art

The NN model of Section III-C3, and here shortly denoted as *NN*, is subjected to a comparative analysis against several previously proposed architectures concerning DLO dynamics prediction. These include the bi-directional LSTM (*BiLSTM*) [12], the interaction-network bi-directional LSTM (*INBiLSTM*) [7], the graph neural network architecture (*GNN*) [10], and the *RBF* network [2]. To ensure a fair comparison, a comparable number of parameters is employed across all the networks. The goals of this section are: i) to compare different neural network-based DLO models, ii) to show that the proposed approach to DLO modeling with conditioning on parameters and online adaptation is architecture-agnostic, iii) to compare the performance of the proposed adaptation of the input model parameters against the direct adjustment of the neural network weights, as in [2].



(a) *white* rope (b) *black* rope (c) *red* rope

Fig. 7. Outcomes of the shape control task involving online adaptation of model parameters, conducted across various rope types and surfaces. Average results across five repetitions per task (standard deviations confidence region intervals).

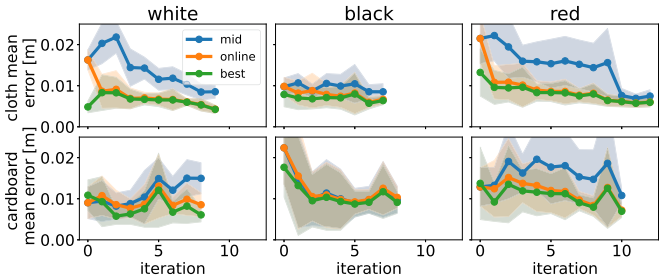


Fig. 8. Comparing prediction errors using mid-range, online, and best model parameters across ropes and surfaces.

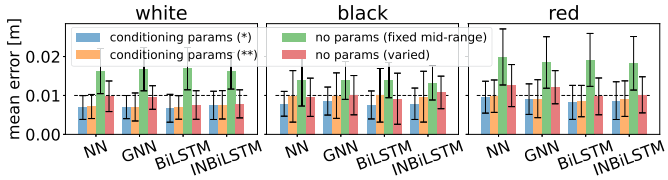


Fig. 9. Prediction error for fix (*no params*) vs conditioning parameters across different models. For the latter, the symbol (*) denotes that the same shape is used for parameters estimation and forward prediction error, whereas with *(**)* the 4-fold cross-validation approach is denoted. With *fixed mid-range* and *varied* the two employed datasets are indicated.

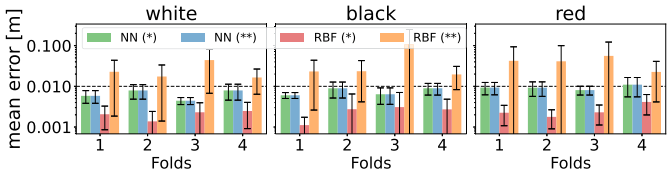


Fig. 10. Mean prediction error (log scale) of the input parameters (ours) vs neural network weights (RBF [2]) update, evaluated on the same shape (*) or on different shapes (**).

The analysis is carried out on the data obtained in the shape control task of Section IV-B on the *cloth* surface, by computing the mean prediction error. A 4-fold cross-validation approach is used. Fold 1 employs the data of the U shape for parameters estimation and the data of the A, S, and I shapes for forward prediction error. The same holds for sets 2, 3 and 4 for shapes A, S and I respectively.

1) Fixed vs Conditioning Parameters: To validate the choice of a NN model conditioned on the model parameters, a new dataset of 275K samples is generated with fixed mid-range parameters ($k_d = 14$, $k_b = 0.5$, mass of 0.02kg, and length in $[0.4, 0.5]$m). The comparison is performed by optimizing the models using three different approaches: 1) the mid-range dataset without considering parameters (*no params (fixed mid-range)*); 2) the varied parameters dataset of Section IV-A without considering parameters (*no params (varied)*); 3) using modified architectures that incorporate model parameters as input (*conditioning params)*. Notably, the latter case implies estimating the parameters according to Section III-D before evaluating the prediction error. This is performed either employing the same shape for both estimation and prediction (*conditioning params (*)*) or with the introduced 4-fold cross-validation procedure (*conditioning params (**)*). The results are shown in Fig. 9. The plots show that conditioning the models on the parameters allows to achieve better accuracy than with the model that is trained to be robust to the range of the parameters (*no params (varied)*), or only with fixed parameters (*no params (fixed mid-range)*), for all considered architectures. Moreover, this is true also when optimized and tested for different shapes (compare *conditioning params (*)* and *(**)* cases, see also Section IV-C2). Additionally, since all architectures show similar accuracy, a simpler and faster fully connected NN model is preferable (see Section IV-C3).

2) Parameters vs Weights Update: Prior work proposed to directly update the weights of the model to achieve online adaptation [2]. In this section, a comparison against this approach is established in Fig. 10, where *fold* refers to the 4-fold cross-validation approach. Fig. 10 shows that updating the network weights directly, as proposed in the *RBF* approach [2], leads to overfitting to the specific target shape, resulting in a loss of generality (compare (*) and (**) cases). In contrast, when the model parameters update is considered, as employed in the proposed approach (NN), a higher level of generalization is observed, resulting in consistent outcomes across various tasks, regardless of the specific task performed during parameter estimation.

3) Architecture Efficiency: To provide an insight about the considered DLO model architecture efficiency, the time to perform forward and backward passes is measured: *NN* $0.20/0.65$ ms; *BiLSTM* $0.48/0.98$ ms; *INBiLSTM* $0.75/1.42$ ms; *GNN* $0.63/1.05$ ms. Taking into account the time complexity of each architecture, it can be concluded that the proposed simple *NN* model emerges as the most favorable.

D. Limitations

The proposed framework exhibits several limitations. First, the action is predicted over a 1-step horizon, in contrast to other approaches [10], [12]. Nevertheless, the current gradient-based action estimation can be expanded to a N-step horizon using the same batch approach. However, encompassing all possibilities would result in an exponential growth of the task. Therefore, the introduction of sampling-based, Top-K, or other techniques is necessary to constrain the prediction task scale. Moreover, our approach is also a forward model of the DLO, so it can be used in any MPC-like framework to enable manipulation planning in longer horizons. The second limitation is related to the fixed number of nodes, e.g. 16 in this work, which would necessitate the generation of new data and the retraining of a new network with adapted dimensions if modified. A third limitation is to assume the DLO dynamics negligible during manipulation. Indeed, the analytical model provides a full trajectory of the DLO response to the pick-and-place action. However, the current NN model only captures the final response.

V. Conclusion

The proposed manipulation framework solves shape control tasks with plane contact involving a range of real-world DLOs and different contact surfaces. The framework exploits an online model parameter estimation procedure to adapt the NN model predictions to the specific DLO being manipulated. The efficiency of the proposed NN model approximating the DLOs' dynamics is exploited both in the actions and parameters estimation routines, and has been proven compared to other architectures. In future works, the proposed NN model will be tested in the context of branched deformable linear objects, i.e. wiring harnesses. Additionally, the extension of the proposed framework to a dual-arm manipulation and to manipulation in presence of environmental obstacles will be investigated.

References

[1] A. Caporali, K. Galassi, B. L. Žagar, R. Zanella, G. Palli, and A. C. Knoll, "RT-DLO: Real-time deformable linear objects instance segmentation," *IEEE Trans. Ind. Informat.*, vol. 19, no. 11, pp. 11333–11342, Nov. 2023.

[2] M. Yu, K. Lv, H. Zhong, S. Song, and X. Li, "Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach," *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 417–436, Feb. 2023.

[3] A. Caporali, M. Pantano, L. Janisch, D. Regulin, G. Palli, and D. Lee, "A weakly supervised semi-automatic image labeling approach for deformable linear objects," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 1013–1020, Feb. 2023.

[4] K. P. Cop, A. Peters, B. L. Žagar, D. Hettegger, and A. C. Knoll, "New metrics for industrial depth sensors evaluation for precise robotic applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5350–5356.

[5] A. Caporali, K. Galassi, and G. Palli, "Deformable linear objects 3D shape estimation and tracking from multiple 2D views," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3852–3859, Jun. 2023.

[6] P. Kicki, A. Szymko, and K. Walas, "DLOFTBs–fast tracking of deformable linear objects with B-splines," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7104–7110.

[7] Y. Yang, J. A. Stork, and T. Stoyanov, "Learning differentiable dynamics models for shape control of deformable linear objects," *Robot. Auton. Syst.*, vol. 158, 2022, Art. no. 104258.

[8] N. Lv, J. Liu, and Y. Jia, "Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2341–2353, Aug. 2022.

[9] N. Lv, J. Liu, X. Ding, J. Liu, H. Lin, and J. Ma, "Physically based real-time interactive assembly simulation of cable harness," *J. Manuf. Syst.*, vol. 43, 2017, pp. 385–399.

[10] C. Wang et al., "Offline-online learning of deformation model for cable manipulation with graph neural networks," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5544–5551, Apr. 2022.

[11] F. Liu, E. Su, J. Lu, M. Li, and M. C. Yip, "Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics," *IEEE Robot. Automat. Lett.*, vol. 8, no. 7, pp. 3964–3971, Jul. 2023.

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

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

[14] W. Zhang, K. Schmeckpeper, P. Chaudhari, and K. Daniilidis, "Deformable linear object prediction using locally linear latent dynamics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 13503–13509.

[15] R. Lagneau, A. Krupa, and M. Marchal, "Automatic shape control of deformable wires based on model-free visual servoing," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5252–5259, Oct. 2020.

[16] P. Kicki, M. Bidziński, and K. Walas, "Learning quasi-static 3D models of markerless deformable linear objects for bimanual robotic manipulation," 2023, *arXiv:2309.07609*.

[17] R. Laezza and Y. Karayiannidis, "Learning shape control of elastoplastic deformable linear objects," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 4438–4444.

[18] T. Tang, C. Wang, and M. Tomizuka, "A framework for manipulating deformable linear objects by coherent point drift," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3426–3433, Oct. 2018.

[19] P. Sundaresan et al., "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 9411–9418.

[20] D. Navarro-Alarcon 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.

[21] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 479–484.

[22] O. Aghajanzadeh, M. Aranda, J. A. Corrales Ramon, C. Cariou, R. Lenain, and Y. Mezouar, "Adaptive deformation control for elastic linear objects," *Front. Robot. AI*, vol. 9, 2022, Art. no. 868459.

[23] A. Khalifa and G. Palli, "Symplectic integration for multivariate dynamic spline-based model of deformable linear objects," *J. Comput. Nonlinear Dyn.*, vol. 17, 2022, Art. no. 011001.