ROBOMECH Journal

RESEARCH

Open Access

# Obstacle avoidance shape control of deformable linear objects with online parameters adaptation based on differentiable simulation

Changjian Ying[1*] and Kimitoshi Yamazaki[2]

**Abstract**

The manipulation of deformable linear objects (DLOs) such as ropes, cables, and hoses by robots has promising applications in various fields such as product assembly and surgical suturing. However, DLOs are more difficult to manipulate than rigid objects because their shape changes during manipulation. Furthermore, preventing a DLO from colliding with the environment is important to prevent it from becoming entangled and causing shape control to fail. In this paper, we proposed an obstacle avoidance and shape control scheme for DLOs based on differentiable simulation that does not require prior data or a specialized controller. First, we established a dynamic model of the DLO that allows for both forward dynamics transfer and error backpropagation to obtain gradients. Then, we employed model predictive control to optimize the embedded neural network for predicting the actions that would manipulate the DLO. Finally, the control scheme was made applicable to DLOs with different material properties by allowing online adaptation of the model parameters essential to deformation during manipulation. Simulations and real-world experiments demonstrate that the proposed control scheme could manipulate the DLO stably and accurately to avoid obstacles and achieve the goal state. In addition, the online adaptation of parameters helped mitigate the sim-to-real gap.

**Keywords** Deformable objects manipulation, Collision avoidance, Differentiable simulation

## Introduction

The robotics community has been extensively studying the manipulation of deformable objects for various applications [1, 2]. Unlike rigid objects, deformable objects change their shape during manipulation due to external forces, which increases the degrees of freedom (DoFs) and makes planning and control of the robot's motion more challenging. Deformable linear objects (DLOs) such as ropes, cables, and hoses are common in households and industry, so tasks involving their manipulation have received increasing attention [3], such as tying knots [4], routing wires [5], and insertion [6]. Furthermore, the DLO manipulation methods can be generalized to deformable objects with more complex geometries such as cloth and soft tissues by extending the dimensionality. Therefore, DLO manipulation is a promising area of study in the robotics field.

This study focuses on the manipulation problem of DLOs: moving a DLO from an initial shape to a given desired shape (i.e., goal shape) while avoiding obstacles. This is referred to as the shape control problem for DLOs. The obstacle avoidance shape control of DLOs has specific applications. For example, in the wiring of an

*Correspondence:
Changjian Ying
21hs252f@shinshu-u.ac.jp
[1] Graduate School of Medicine, Science and Technology, Shinshu University, Nagano, Japan
[2] Faculty of Engineering, Shinshu University, Nagano, Japan

automobile interior, the limited space and large number of parts inside the automobile create a challenging environment for the cable, which is susceptible to entanglement or severe damage. Therefore, one of the essential applications of our proposed method is to develop a solution that can dexterously avoid contact with other parts to achieve the desired shape. In our previous work [7], we proposed a differentiable simulation framework for modeling DLO deformation and generating obstacle avoidance trajectories offline. In this study, we extended the previous method into a shape control and obstacle avoidance scheme for DLO manipulation with online adaptation of the model parameters to mitigate the sim-to-real gap. In contrast to our previous method, the proposed control scheme does not rely on prior real-world data to determine the model parameters.

Solving the shape control problem requires developing an accurate model of DLO deformation. Specifically, the relation $\boldsymbol{s} = \mathcal{F}(\boldsymbol{a})$ between the state $\boldsymbol{s}$ of the DLO and action $\boldsymbol{a}$ needs to be mapped, where $\mathcal{F}$ is a blackbox function that outputs the state of the DLO (i.e., position or velocity) when an action (i.e., force or velocity) is input. A rigid body has few DoFs, which allows for an accurate formulation of $\mathcal{F}$ based on physical laws. The inverse kinematics can then be solved by using $\boldsymbol{a} = \mathcal{F}^{-1}(\boldsymbol{s})$ to control the state of the object. However, DLOs have many more DoFs than a rigid body and the action applied to the DLO has much fewer DoFs than the state of the DLO, so establishing such an underactuated control system is a major challenge.

Analytical models and data-driven methods are often used to model the deformation of a DLO. In computer graphics, the analytical model of a DLO is relatively straightforward compared with that of more complex deformable objects [8]. In robotics, the deformation of DLOs is commonly modeled by using mass–spring systems [9], the finite-element method (FEM) [10], and position-based dynamics (PBD) [11] because of their ease of implementation. However, these analytical models are only approximate, and they assume that the material of the DLO remains isotropic. The sim-to-real gap problem can be defined as the need to determine certain model parameters so that the model can achieve a close approximation of reality. Traditional methods for optimizing model parameters involve minimizing the error between simulated and real data. However, this approach can be time-consuming, and collecting new data when the material properties of the DLO have been altered is impractical. Therefore, the data-driven (i.e., learning-based) approach [12–14] is commonly used to model the deformation of a DLO. Learning-based models such as neural networks (NNs) are trained to approximate the dynamics of a DLO. Then, techniques such as transfer learning and online learning can be used to apply trained models to untrained objects with different material properties. However, this approach requires the collection of large amounts of prior data for training, which usually comes from commercial physics engines such as Unity, Mujoco, and Isaac Sim instead of collecting real-world data. Furthermore, learning-based models have difficulty with effectively learning the dynamics of a DLO in contact with the environment [15] or can only simulate some simple responses to a collision [16]. Thus, few learning-based models can consider scenarios with obstacles, which are ubiquitous in manufacturing sites.

In this paper, we consider the obstacle avoidance shape control problem for DLO, meaning that in addition to obtaining the deformation model of DLO a proper control scheme must be established. Given the sim-to-real gap of analytical models and large amount of prior data required by data-driven models, an alternative approach is to use differentiable simulation, which allows the deformation of DLOs to be simulated by using an analytical model without requiring any prior data. Moreover, differentiable simulation offers end-to-end differentiable capability that can be used to optimize the model parameters related to shape control online to reduce the sim-to-real gap. Specifically, we implemented an analytical model of DLO based on PBD from scratch in a differentiable framework. To obtain analytic gradients with respect to the parameters that need to be optimized, we made necessary modifications to the analytical model of DLO due to the limitations of the auto-differentiation criterion. These gradients can then be used to update the parameters using gradient-based optimization methods. The differentiable simulation seamlessly embedded an NN controller, which predicts the actions necessary for shape control of the DLO. Model Predictive Control (MPC) optimizes the NN controller to output correct obstacle avoidance actions, even if the goal and environment configuration changes. Additionally, we utilized a markerless perception algorithm to effectively recognize the state of the DLO during manipulation, thereby mitigating sim-to-real errors caused by inappropriate simulation parameters through closed-loop visual feedback.

The contributions of this study are as follows:

- We extended the existing simulator to compute ready-to-use gradients from simulations that incorporate collision effects.
- We proposed a control scheme for the DLO to reach the goal state while avoiding obstacles. The control scheme can be adapted to different types of DLOs by adjusting deformation-related model parameters to reduce the sim-to-real gap.

- Simulation and real-world experiments were conducted to evaluate the effectiveness and stability of the proposed control scheme.

The remainder of this paper is structured as follows. Sect. "Related work" presents a literature review on related studies. Sect. "Problematization and approach" describes the problem setting and presents an overview of the proposed control scheme. Sects. "Differentiable position-based dynamics" and "Shape control with online parameters adaptation" describe the parts of the proposed control scheme in detail. Sect. "Validation" presents the simulation and real-world experiments performed to validate the proposed control scheme. Sect. "Conclusion" concludes the paper.

## Related work

DLO manipulation can be categorized as a control or planning problem. The former is usually limited to small deformation tasks and does not consider the effects of contact with the environment whereas the latter considers large deformation tasks in a constrained environment. Below, we review some common approaches used to solve these two problems.

### Analytical models

Analytical models are a mature approach and are widely used to control or optimize the actions of a DLO. Such models are highly interpretive and can be easily integrated into classical control theory. Among analytical models, FEM [10, 17, 18] based on strict material and mechanical laws is generally considered the most accurate. However, high precision in simulations comes with drawbacks such as high computational requirements and difficult implementation. Moreover, the simulation results can be heavily influenced by material properties such as Young's modulus and Poisson's ratio. To address this issue, some researchers have resorted to empirical determination or derivative-free optimization algorithms, such as the Nelder–Mead algorithm [19] and grid search [20]. However, these techniques require collecting prior data again to determine the material properties whenever the properties of the DLO change. Alternatively, mass–spring systems and PBD have been used to approximate exact physical phenomena to simulate DLO deformation [21, 22]. Although these methods are easy to implement and fast, they encounter the same difficulty with determining critical parameters such as stiffness coefficients.

For DLOs, the planning problem involves generating a path from the initial state to the goal state while avoiding obstacles. Because of the high dimensionality of a DLO, obtaining an analytical solution for inverse kinematics is impractical. Therefore, sampling algorithms such as the rapidly exploring random tree (RRT) [23] and probabilistic roadmap methods (PRM) [24] have become widely adopted. Researchers then integrate an analytical model capable of generating valid DLO shapes into a classical high-dimensional planner [25–28]. However, such methods heavily rely on an accurate DLO energy model to calculate stable shapes in equilibrium. In addition, the absence of closed-loop control means that there is no assurance of a favorable execution outcome in the real world.

### Learning-based methods

The application of learning-based methods to DLO manipulation can be divided into three main categories: offline learning using NNs to approximate the global dynamics of a DLO, online inference of Jacobian matrices representing the local deformations of a DLO, and end-to-end learning of a control strategy.

For offline learning of the global dynamics, various network constructs have been proposed. Wang et al. [13] and Li et al. [29] used graph neural networks to treat a DLO as a graphical structure with vertices and edges. Similarly, Yang et al. [12] and Yan et al. [30] used bidirectional long short-term memory to learn the interactions between the segments of a DLO. To avoid the difficulty of directly learning nonlinear dynamics, Zhang et al. [31] and Yan et al. [32] encoded the image of a DLO into a latent space where linear dynamics can be easily learned. Each of these approaches uses model predictive control (MPC) to control the differentiability of actions to manipulate the DLO because NNs are inherently differentiable. However, obtaining differentiable dynamics requires first collecting a large amount of prior data from the real world or simulations. In addition, learning the complex dynamics of a DLO in contact with its environment remains an issue.

In online inference, small local deformations of the DLO are assumed linear. This assumption can be used to create a Jacobian matrix that maps the velocity of the robot's end-effector to the states of the DLO. The control inputs applied by the robot to the DLO can then be derived by solving the inverse Jacobian matrix, which is commonly referred to as visual servoing [33–35]. Berenson et al. [36] approximated the Jacobian matrix by using the diminishing stiffness of a deformable object. Zhu et al. [37] parameterized the shape of the DLO by using a Fourier series and used the shape parameter to estimate the Jacobian matrix of the local deformations of a cable for online two-handed manipulation. Yu et al. [14] used a radial basis function network to learn the Jacobian matrix of local deformations from simulation data. They then updated the Jacobian matrix online to compensate for errors caused by changes in the properties of the DLO

and inaccurate modeling. However, these methods have limited ability to handle large deformations or account for the presence of obstacles because they constrain nonlinear DLO deformations to linear deformations.

Reinforcement learning (RL) is typically used for end-to-end learning of a DLO control strategy [38], where the actions of a robot are directly output from visually observed inputs. Han et al. [39] used the model-based RL algorithm PILCO to learn a strategy for manipulating a DLO on a 2D plane. Wu et al. [40] used the model-free RL algorithm soft actor–critic (SAC) to learn a strategy for placement and selection where images are taken directly as the input. However, RL methods are typically trained on simulation data because of the high cost of obtaining real-world data. Therefore, further research is needed on whether the learned strategies perform well in the real world. Additionally, imitation learning has recently garnered considerable attention due to its ability to enable robots to generate motions based on human demonstrations at speeds comparable to human performance. Examples include teaching a robot to use a tool to pick up objects [41] or perform long-term tasks such as writing and erasing [42]. However, a limitation of these studies is that generalization to untrained tasks does not yield stable results or requires the re-collection of teaching data for new tasks. In contrast, methods based on differentiable simulation can generalize the control scheme to various tasks by modifying the objective function, without the necessity of reconstructing the model.

### Differentiable simulation

In the field of robot manipulation, differentiable simulation is primarily used for parameter identification [43] and motion control [44, 45]. The main distinction between differentiable simulation and analytical models is that the former requires an analytical model of the dynamics to be constructed as well as backpropagation of errors like in deep learning to obtain the analytical gradients of the model parameters. These gradients can then be used to optimize the parameters. Thanks to recent advances in differentiable programing, automatic differentiation techniques can now be applied to the error backpropagation of analytical models. Various studies have applied differentiable simulation to manipulating deformable objects. Chen et al. [46] proposed an end-to-end learning pipeline that uses a differentiable physics engine to teach a NN how to represent high-dimensional point cloud data collected from a deformable object. Millard et al. [47] represented a deformable object with a tetrahedral FEM mesh and optimized the material parameters to minimize the discrepancy between the observed and predicted values. Liu et al. [21] proposed differentiable formulations to represent the deformations of extensible and inextensible rope-like objects. These formulations solve the parameter estimation problem and mitigate the sim-to-real gap to match rope physics to realistic scenarios. Our approach is similar to that of Liu et al. [21], but we also made collision effects differentiable during the simulation episode and achieved obstacle avoidance shape control, which is not considered in [21]. In addition, we achieved online parameter adaptation coupled with shape control, rather than offline parameter identification as presented in [21].

## Problematization and approach

### Problem setting

This study addresses the problem of dynamic manipulation planning for deformable objects, without considering the manipulation sequence between different objects such as grasping, moving, and dropping. In particular, a DLO is continuously moved from an initial state to a goal state within a constrained environment. The manipulations must be dynamically planned to prevent collisions between the object and the environment throughout the movement. This category of problems involves deformable objects that undergo continuous state changes and deformations during manipulation, necessitating real-time sensing and prediction of their shape and position. To characterize this, we created a laboratory scenario as shown in Fig. 1. The state of the DLO is represented by the positions of several nodes uniformly distributed along its length. During manipulation, the DLO must avoid obstacles and overstretching (i.e., the distance between the two endpoints cannot exceed the original length of the DLO). Our aim was to develop a control scheme that generates a sequence of actions by the end-effectors of the robot that minimizes the positional differences between the DLO after manipulation and the goal state. We made the following assumptions:

- The robot does not move very quickly, and the DLO is in a quasi-static state, which means that the DLO

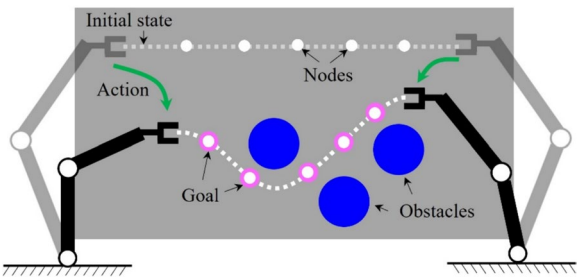



**Fig. 1** Schematics of the DLO manipulation task

is only subjected to elastic and gravitational potential energies while inertial effects can be ignored.
- The DLO is made from an isotropic and inextensible material that only undergoes elastic deformation.
- An RGB-D camera captures the entire state of the DLO, and there is no occlusion or self-intersection during manipulation.
- The positions and shapes of the obstacles are fixed and known.

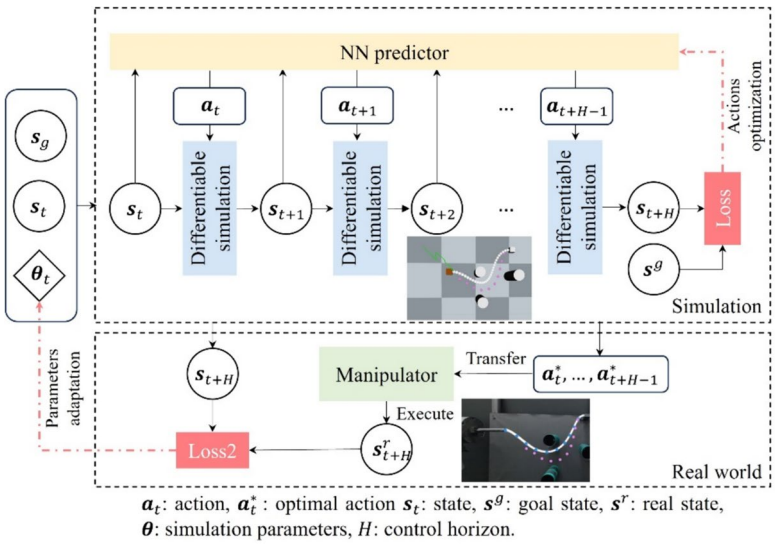
Then the manipulation of the DLO can be described as an optimization problem:

$$\begin{aligned} \boldsymbol{a}^{*}_{1:T-1} &= \arg\min_{\boldsymbol{a}} \left( loss\left(\boldsymbol{s}_T, \boldsymbol{s}^g\right)\right) \\ s.t.\ \ \boldsymbol{s}_{2:T} &= D_{\boldsymbol{\theta}}(\boldsymbol{s}_1, \boldsymbol{a}_{1:T-1}) \#(1) \end{aligned} \tag{1}$$

Here, $\boldsymbol{s}_{1:T} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T] \in \mathbb{R}^{3nT}$ denotes all states of the DLO in the timespan $T$, where $\boldsymbol{s}_t = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{3n}$ denotes the state of the DLO at the time step $t$, $\boldsymbol{x}_k \in \mathbb{R}^3$ denotes the $k$ th node of the DLO, and n is the number of nodes. $\boldsymbol{a}_{1:T-1} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{T-1}] \in \mathbb{R}^{\mu(T-1)}$ denotes the sequence of actions throughout the timespan, $\boldsymbol{a}_t \in \mathbb{R}^{\mu}$ denotes the action at the time step $t$, and $\mu$ is the DoF of the action. The function $\mathcal{D}_{\boldsymbol{\theta}}$ represents the dynamics of the DLO, where $\boldsymbol{\theta}$ denotes parameters related to the deformation. When the state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$ of the current time step $t$ are given, the state of the next time step $\boldsymbol{s}_{t+1}$ can be computed as $\boldsymbol{s}_{t+1} = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The function *loss* denotes the objective function used to solve the optimization problem, $\boldsymbol{s}_1$ and $\boldsymbol{s}_T$ denote the initial and final states of the DLO before and after manipulation, respectively, and $\boldsymbol{s}^g = [\boldsymbol{x}^g_1, \ldots, \boldsymbol{x}^g_n] \in \mathbb{R}^{3n}$ denotes the goal state. Solving this optimization problem generates an optimal sequence of actions $\boldsymbol{a}^{*}_{1:T-1}$ that minimizes the error between the final and goal states.

### Outline of the proposed control scheme

Figure 2 shows the proposed control scheme based on differentiable simulation. This approach has two advantages: the dynamics of the DLO can be obtained without relying on prior data, and the scheme can be adapted to different types of DLOs. First, PBD is used to construct an analytical model that represents the deformation of the DLO with end-to-end differentiability. The use of automatic differentiation eliminates the need for a data-driven approach to obtain the dynamics of the DLO while providing the required gradients for task optimization. Next, an NN is integrated into the model as a controller to predict the corresponding action in the current state. MPC is used to update the NN weights via a gradient-based optimization algorithm to determine the optimal action. The upper half of Fig. 2 illustrates one control horizon of the MPC given the current state $\boldsymbol{s}_t$ and model parameter $\boldsymbol{\theta}_t$. By minimizing the error between the goal state $\boldsymbol{s}^g$ and the state $\boldsymbol{s}_{t+H}$ of the DLO after a control horizon, an optimized action sequence $\left[\boldsymbol{a}^{*}_t, \ldots, \boldsymbol{a}^{*}_{t+H-1}\right]$ can be obtained. To reduce the sim-to-real gap, we can adjust the parameters associated with deformation of the DLO during execution of the optimized actions. As shown in the lower half of Fig. 2, this is done by minimizing the error between the simulated state $\boldsymbol{s}^r_{t+H}$ and corresponding observed state $\boldsymbol{s}_{t+H}$ of the DLO. Thus, the proposed control scheme can be applied to different




$\boldsymbol{a}_t$: action, $\boldsymbol{a}^{*}_t$ : optimal action $\boldsymbol{s}_t$: state, $\boldsymbol{s}^g$: goal state, $\boldsymbol{s}^r$: real state, $\boldsymbol{\theta}$: simulation parameters, $H$: control horizon.

**Fig. 2** Proposed control scheme: one control horizon for MPC (upper) and online parameter adaptation (lower)

types of DLOs without the need to collect prior data from the real world.

In a related approach, Yang et al. [48] proposed an algorithm for action control based on learning dynamics. In contrast, our proposed control scheme obtains the gradients from the simulation directly followed by action optimization and parameter identification. This circumvents the need to train the NN and interpret the rationality of the network structure. This work is an extension of our previous work [7], where we proposed a method of generating obstacle avoidance trajectories to reach the goal state offline. Our proposed control scheme improves upon our previous work in three main aspects: we use MPC to shift shape control of the DLO from an open loop to a closed loop, we integrate a parameter identification module into the control scheme, and we validated our proposed control scheme by conducting additional simulations and real-world experiments.

## Differentiable position-based dynamics

### Motivation

We modeled the dynamics of a DLO by using Cosserat rods [49], which we implemented by using PBD. To reduce the computational cost, we used compliant PBD (XPBD) [50]. Traditional dynamic models calculate the acceleration of an object based on the applied force and then updates the velocity and position of the object by integration over time. In contrast, PBD directly modifies the positions of the particles (which are equal to the nodes of the DLOs mentioned in the problem setting) that compose the object and thus does not need to solve for velocity, which makes the model lightweight and easy to implement.

We selected Cosserat rods [49] to model the dynamics of the DLO for several reasons. First, this model can accurately represent elastic deformations such as stretching, shearing, bending, and twisting of the DLO. In addition, constraints can be incorporated to simulate responses to a collision. Second, external manipulations of the DLO can be directly represented as a sequence of positions, which avoids the problem of overshooting due to explicit integration. Third, the computational formulas are fully differentiable, which facilitates differentiable simulation in an auto-differentiation framework without requiring additional approximations such as the linear complementarity problem. PBD deviates from real-world behavior because it prioritizes visual plausibility over strict adherence to physical laws, but the sim-to-real gap can be narrowed by online adaptation of the model parameters. For brevity, only the key formulas are presented here. Detailed derivations are available in the literature [49, 50]. We adopted Kugelstadt and Schömer's concept [49] of treating the orientation as the same as the position and their stretch/shear constraints and bending/twisting constraints. We made three extensions to their original model: we added a collision constraint to simulate deformations that occur when the DLO contacts with obstacles, we employed XPBD to solve the constraint equations, and we implemented an auto-differentiation framework to obtain the gradients.

### Original model

Figure 3 shows the Cosserat rods used to model a DLO with $n$ positions and $n+1$ orientations. The positions are defined as $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots \boldsymbol{x}_n] \in \mathbb{R}^{3n}$ in a Cartesian coordinate system. The orientations are defined as $\boldsymbol{q} = \left[\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{n+1}\right] \in \mathbb{R}^{4(n+1)}$, where $\boldsymbol{q}_i \in \mathbb{R}^4$ is a quaternion that represents the rotation from the global frame $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3) \in \mathbb{R}^3$ rotated to the material frame. To simplify the notation, the position and orientation are integrated into the set $\boldsymbol{p} = [\boldsymbol{x}, \boldsymbol{q}]$. The corrections to the position and orientation $\Delta \boldsymbol{p}$ are calculated for each time step by solving the set of constraint equations $\boldsymbol{C}(\boldsymbol{p} + \Delta \boldsymbol{p}) = 0$. These constraints can be linearized by using the Taylor series expansion:

$$C(\boldsymbol{p} + \Delta \boldsymbol{p}) \approx C(\boldsymbol{p}) + \nabla_{\boldsymbol{p}} C \Delta p = 0 \tag{2}$$

where $\nabla_{\boldsymbol{p}} \boldsymbol{C}$ is the Jacobian matrix of $\boldsymbol{C}$ with regard to vector $\boldsymbol{p}$. This equation can be solved by restricting $\Delta \boldsymbol{p}$ to the derived orientation of the constraint function:

$$\Delta p = \boldsymbol{M}^{-1} \left(\nabla_{\boldsymbol{p}} \boldsymbol{C}\right)^T \Delta \lambda, \tag{3}$$

where $\boldsymbol{M}$ is the mass/inertia matrix $diag(m_1 \cdot 1, m_2 \cdot 1, \ldots, m_n \cdot 1, \boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_{n-1})$. The change in the Lagrange multiplier $\Delta \boldsymbol{\lambda}$ can be computed separately for each constraint $j$ by using the Gauss–Seidel solution.

The original model defines two constraints to represent the deformability of a DLO, which we describe here.

*Stretch and shear constraint* Based on the Cosserat theory, the measured strain due to shear and stretching was

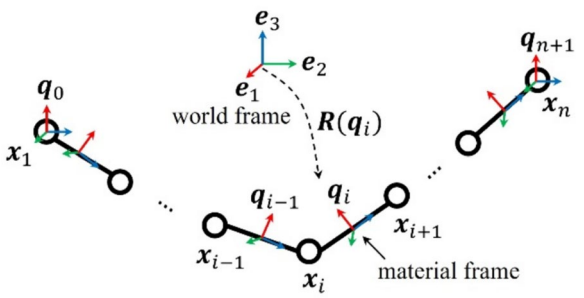



**Fig. 3** Geometry of Cosserat rods represented as positions and orientations

coupled to the difference between the tangent vector of the centerline and the normal of the cross-section. Therefore, the stretch and shear constraint $C_s$ is formed by two adjacent particles with positions $x_i$ and $x_{i+1}$ and the quaternion $q_i$ between them:

$$\boldsymbol{C}_s(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \boldsymbol{q}_i) = (\boldsymbol{x}_i - \boldsymbol{x}_{i+1})/l - R(\boldsymbol{q}_i)\boldsymbol{e}_3, \tag{4}$$

where $l$ is the resting length between the two particles. For simplicity, we assumed that all $l$ values are equal. $\boldsymbol{R}(\cdot)$ denotes the rotation matrix converted from the quaternion. Next, the derivatives with regard to the involved positions and quaternions can be obtained:

$$\nabla_{\boldsymbol{x}_i}\boldsymbol{C}_s = -\nabla_{\boldsymbol{x}_{i+1}}\boldsymbol{C}_s = -1/l \cdot 1_{3\times 3}, \tag{5}$$

$$\nabla_{\boldsymbol{q}}\boldsymbol{C}_s = 2\big(\Im(\boldsymbol{e}_3\overline{\boldsymbol{q}})|\Re(\boldsymbol{e}_3\overline{\boldsymbol{q}})1_{3\times 3} - [\Im(\boldsymbol{e}_3\overline{\boldsymbol{q}})]^{\times}\big), \tag{6}$$

where $\Im(\cdot)$ and $\Re(\cdot)$ denote the imaginary and real parts, respectively, of the quaternion, $\overline{\boldsymbol{q}}$ is the conjugate quaternion, and $[\cdot]^{\times}$ denotes the skew-symmetric matrix of a vector.

*Bend and twist constraint* We employed the Darboux vector $\boldsymbol{\Omega}$ to define the measured strain due to bending and twisting. The bend and twist constraint $\boldsymbol{C}_b$ can be coupled with the two adjacent quaternions $\boldsymbol{q}_i$ and $\boldsymbol{q}_{i+1}$ to compute the difference from the resting value:

$$\boldsymbol{C}_b(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}) = I\big(\overline{\boldsymbol{q}_i}\boldsymbol{q}_{i+1} - \overline{\boldsymbol{q}_i}^0\boldsymbol{q}_{i+1}^0\big) = \Omega - \xi\boldsymbol{\Omega}^0, \tag{7}$$

$$\xi = \begin{cases} +1 if \left|\boldsymbol{\Omega} - \boldsymbol{\Omega}^0\right|^2 < \left|\boldsymbol{\Omega} + \boldsymbol{\Omega}^0\right|^2, \\ -1 if \left|\boldsymbol{\Omega} - \boldsymbol{\Omega}^0\right|^2 > \left|\boldsymbol{\Omega} + \boldsymbol{\Omega}^0\right|^2, \end{cases} \tag{8}$$

where $\boldsymbol{q}_i^0$ and $\boldsymbol{\Omega}^0$ denote the quaternion and Darboux vector at the resting state, respectively. The derivatives from the involved quaternions were obtained as follows:

$$\nabla_{\mathbf{q}_i}\boldsymbol{C}_b = -\Big(-\boldsymbol{q}_{i+1}|q_{i+1,0}1_{3\times 3} - \big[\boldsymbol{q}_{i+1}\big]^{\times}\Big), \tag{9}$$

$$\nabla_{\mathbf{q}_{i+1}}\boldsymbol{C}_b = +\Big(-\boldsymbol{q}_i|q_{i,0}1_{3\times 3} - \big[\boldsymbol{q}_i\big]^{\times}\Big). \tag{10}$$

### Extensions for our approach

We added a third constraint to represent the deformability of a DLO.

*Collision constraint* The response of the DLO to a collision with an obstacle must be considered. We did not consider the response of the DLO to collision with itself to reduce the computation time. In PBD, the collision response can also be treated as a constraint. First, we can test the ray $\boldsymbol{x}_{t,i} \rightarrow \boldsymbol{x}_{t+1,i}$ for each particle $i$ to determine whether the ray penetrates the obstacle, and we can compute the entry point $\boldsymbol{x}_{ic}$ and surface normal $\boldsymbol{n}_{ic}$ at this position. $\boldsymbol{x}_{ic}$ is replaced by the point on the surface of the obstacle nearest to $\boldsymbol{x}_{t,i}$ if the ray lies completely inside the obstacle, and $\boldsymbol{n}_{ic}$ is replaced by the normal at the nearest point. Next, the unilateral constraint function $\boldsymbol{C}_c(\boldsymbol{x}_i) = (\boldsymbol{x}_i - \boldsymbol{x}_{ic}) \cdot \boldsymbol{n}_{ic}$ is defined, and an inverse stiffness $\beta = 1$ is added. Then, the derivative with regard to the position can be readily obtained by $\nabla_{\boldsymbol{x}_i}\boldsymbol{C}_c = \boldsymbol{n}_{ic}$.

XPBD can be to solve the constraint Eq. (3):

$$\Delta\lambda_j = -\Big(\nabla_{\boldsymbol{p}}C_j\boldsymbol{M}^{-1}\nabla_{\boldsymbol{p}}C_j^T + \widetilde{\beta}_j\Big)^{-1}\Big(C_j(\boldsymbol{p}_i) + \widetilde{\beta}_j\lambda_{ij}\Big). \tag{11}$$

Here, $\lambda_{ij}$ is the total Lagrange multiplier for constraint $j$ at the current iteration $i$. $\widetilde{\boldsymbol{\beta}} = \boldsymbol{\beta}/\Delta t^2$, where $\boldsymbol{\beta}$ and $\Delta t$ denote the inverse stiffness and time step, respectively.

**Algorithm 1** Modified PBD for differentiable simulation

```
initialize state x,q,λ
initialize the gradient variable θ←{a,m,I,ξ_v,ξ_ω,β_b}
while t < timeSteps do
  for all particles i do
    x_{t,i}←x_{t−1,i} + Δt v_{t,i} +Δt² M⁻¹ f_ext
  end for
  for all manipulated particles s do x_{t,s}←a_{t,s}
  while j < solverIterations do
    compute Δλ using Eq. (5)
    compute Δp using Eq. (4)
    λ_{t,j}←λ_{t,j−1} + Δλ
    p_{t,j}←p_{t,j−1} + Δp
    j←j +1
  end while
 t←t + 1
end while
ℒ←loss(x,q)
∇_θ ℒ←backpropagation(ℒ)
return x, ∇_θ ℒ
```

After solving all the constraint functions, the linear and angular velocities of each particle are updated by the corrections to the position and quaternion. Then, the velocity is multiplied by a damping factor, such as, $v = (1 - \xi_v)^{\Delta t}v$, to represent the energy dissipation. The damping factor $\xi_v$ indicates the damping ratio per second of the corresponding velocity. The larger the damping factor, the more energy is dissipated and the faster the model is stabilized.

The original PBD updates the position and orientation at each time step but only stores the most recent state and overwrites past states. Differentiable simulation requires storing all intermediate states, which allows automatic differentiation tools to compute the analytic gradients of

the parameters. Thus, Algorithm 1 presents the modified PBD framework for differentiable simulation. Lines 1 and 2 initialize the state of the object and create the parameter sequence required to compute the gradients, such as the stiffness $\boldsymbol{\beta}$ and action $\boldsymbol{a}$. Lines 3–14 present the standard PBD framework. To calculate the gradients, all momentary states and the intermediate process of iteratively solving the constraints are stored in lines 8 and 9. Lines 15 and 16 compute the error according to a predefined objective function. Then the gradient of the error with respect to the parameter $\boldsymbol{\theta}$ is computed by backpropagation.

## Shape control with online parameters adaptation

### Model predictive control

The shape control of a DLO can be treated as an optimization problem for MPC, where the entire timespan $T$ is divided into several control horizons $H$ and the actions in each control horizon are optimized. For a given state $\boldsymbol{s}_t$ at the time step $t$, a goal state $\boldsymbol{s}^g$, and the dynamics $\mathcal{D}$ of the DLO, a controller is required that can predict the action $\boldsymbol{a}_t$ at the current time step, which we approximated by using an NN with four fully connected layers. The input layer with a size of $n \times 3 \times 2$ neurons consists of the current state $\boldsymbol{s}_t$ and goal state $\boldsymbol{s}^g$. The two hidden layers both have a size of 256 neurons, and the output layer with a size of $\mu$ neurons consists of the movement velocity $\boldsymbol{u}_t \in \mathbb{R}^\mu$. Hidden layers are connected by a ReLU function, other layers are connected by linear functions. The prediction of the NN is represented by $\pi_{\boldsymbol{w}}(\boldsymbol{s}_t, \boldsymbol{s}^g) = \boldsymbol{u}_t$, where $\boldsymbol{w}$ denotes the weight of the NN. The current state can be computed by using $\boldsymbol{a}_t = \boldsymbol{a}_{t-1} + \Delta t \boldsymbol{u}_t$, where $\Delta t$ is the time interval. The NN is embedded into the model as shown in Fig. 2 to optimize the weights using a gradient-based optimization method.

We adopted a shooting approach for MPC. For a given control horizon, the current state $\boldsymbol{s}_t$ is input to the NN, which outputs the current action $\boldsymbol{a}_t$ according to $\boldsymbol{a}_t = \boldsymbol{a}_{t-1} + \pi_{\boldsymbol{w}}(\boldsymbol{s}_t, \boldsymbol{s}^g) \cdot \Delta t$. The state in the next time step $\boldsymbol{s}_{t+1}$ can then be computed based on the dynamics of the DLO: $\boldsymbol{s}_{t+1} = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. By analogy, the state sequence $\boldsymbol{s}_{t+1:t+H} = [\boldsymbol{s}_{t+1}, \ldots, \boldsymbol{s}_{t+H}]$ and action sequence $\boldsymbol{a}_{t:t+H-1} = [\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+H-1}]$ can be computed. The weights of the gradients with respect to the NN are backpropagated from the sum of future errors. To avoid obstacles, we define a specialized objective function $\mathcal{L}_1$ as illustrated in Fig. 4 that applies an artificial potential field:

$$\mathcal{L}_1(\boldsymbol{s}_{t+1:t+H}, \boldsymbol{s}^g) = \frac{1}{H} \sum_{i=t+1}^{t+H} \left( \omega_1 \|\boldsymbol{s}_i - \boldsymbol{s}^g\| + \omega_2 U(\boldsymbol{s}_i) + \omega_3 \|\boldsymbol{u}_i\|^2 \right) \tag{12}$$

where $\omega_1$, $\omega_2$ and $\omega_3$ are the tradeoff coefficients. They represent the scaling of the corresponding terms.

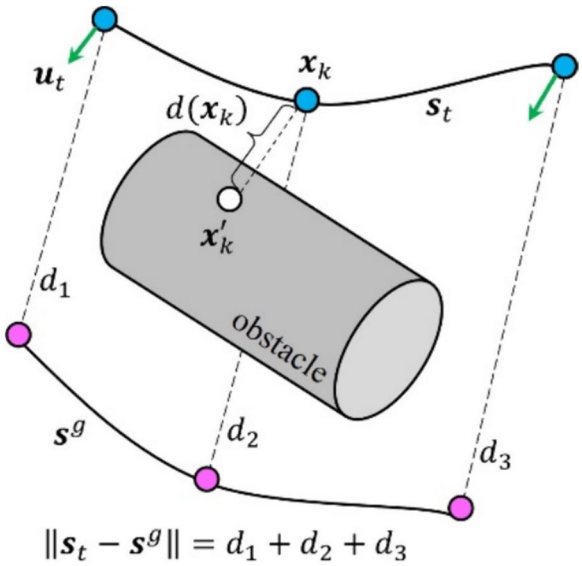



**Fig. 4** Definitions of the terms in the objective function $\mathcal{L}_1$

$\|\boldsymbol{s}_i - \boldsymbol{s}^g\|$ represents the positional error between the current state and goal state, and $\|\boldsymbol{u}_i\|^2$ is a term that limits the speed of the action. $U(\boldsymbol{x}_k)$ is designed with a repulsive potential to prevent the node $\boldsymbol{x}_k$ from colliding with the obstacle $\mathcal{O}$:

$$U(\boldsymbol{x}_k) = \begin{cases} \left( \frac{1}{d(\boldsymbol{x}_k)} - \frac{1}{\varepsilon_d} \right)^2, & d(\boldsymbol{x}_k) < \varepsilon_d \\ 0, & otherwise \end{cases} \tag{13}$$

where $d(\boldsymbol{x}_k) = \min_{\boldsymbol{x}_{k'} \in \mathcal{O}} \|\boldsymbol{x}_k - \boldsymbol{x}_{k'}\|$ is the closest distance between a node on the DLO and the obstacle and $\varepsilon_d$ is the maximum distance at which the repulsive potential works. Then, each control horizon can be represented as an optimization problem for the weights of the NN:

$$\begin{aligned} \boldsymbol{w}^* &= \arg\min_{w} \mathcal{L}_1(\boldsymbol{s}_{t+1:t+H}, \boldsymbol{s}^g) \\ s.t. \boldsymbol{s}_{t+1} &= \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t) \\ \boldsymbol{a}_t &= \boldsymbol{a}_{t-1} + \Delta t \pi_{\boldsymbol{w}}(\boldsymbol{s}_t, \boldsymbol{s}^g) \end{aligned} \tag{14}$$

The gradient optimization algorithm efficiently solves the optimization problem by using the gradients obtained by backpropagation. Combining differentiable simulation with MPC offers two advantages. First, there is no need to learn gradients by learning the dynamics from prior data. Second, compared to model-free RL algorithms, various similar tasks can be accomplished by simply changing the objective function without retraining.

**Algorithm 2** Shape control with online adaptation of parameters

```
   input time steps T, horizon H, iteration ℐ
1  initialize 𝒟_θ, π_w
2  while t < T do
3    for i to ℐ do
4      for j to H do
5        compute a_{t+j} using π_w
6        compute s_{t+j} using 𝒟_θ
7      end for
8      compute ℒ_1 using Eq. (13)
9      obtain ∇_w ℒ_1 from Algorithm 1
10     update w using ∇_w ℒ_1
11   end for
12   execute a*_{t:t+H−1} in real world
13   perceive s^r_{t+H} from real world
14   for i to ℐ do
15     execute a*_{t:t+H−1} in simulation
16     ℒ_2 ← ‖s_{t+H} − s^r_{t+H}‖
17     obtain ∇_θ ℒ_2 from Algorithm 1
18     update θ using ∇_θ ℒ_2
19   end for
20   t ← t + H
21 end while
```

### Online adaptation of parameters

The simulated deformation of a DLO is influenced by physical parameters such as the stiffness, mass, and damping. To narrow the sim-to-real gap, we can modify such parameters based on visual feedback from shape control of the DLO. In our proposed control scheme, the completion of a control horizon generates the state sequence $\boldsymbol{s}_{t+1:t+H}$ and action sequence $\boldsymbol{a}_{t:t+H-1}$. Executing the generated action sequence in the real world generates the state sequence $\boldsymbol{s}^r_{t+1:t+H}$ in the real world, which can be obtained via a visual perception algorithm. The adaptation of the model parameters can then be represented as an optimization problem:

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\min_{\theta} \mathcal{L}_2\left(\boldsymbol{s}_{t+H}, \boldsymbol{s}^r_{t+H}\right) \\ & s.t. \boldsymbol{s}_{t+1} = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, \boldsymbol{a}_t) \end{aligned} \tag{15}$$

where $\boldsymbol{\theta}$ denotes the model parameter that needs to be recognized and $\mathcal{L}_2 = \|\boldsymbol{s}_{t+H} - \boldsymbol{s}^r_{t+H}\|$ represents the positional error between the simulated and real-world states. To save manipulation time, only the final state in one control horizon is used to calculate the error.

Algorithm 2 incorporates online parameter adaptation represented by (15) into the shape control discussed in Sect. "Model predictive control". Line 1 initializes the model parameters and weights of the NN. Line 20 indicates that the entire timespan is divided into several control horizons $H$ for processing. Lines 4–11 describe the optimization process of the MPC for one control horizon, from which a sequence of control horizon actions $\boldsymbol{a}^*_{t:t+H-1}$ is generated. Lines 12 and 13 execute $\boldsymbol{a}^*_{t:t+H-1}$ in the real world, which moves the DLO to an intermediate state. Then, the real-world state of the DLO is obtained based on the visual perception algorithm. Lines 14–19 repeatedly execute the same action sequence in the simulation and iteratively update the model parameters by comparing the simulated and real-world states of the DLO.

## Validation

### Simulations

We conducted simulations to investigate the accuracy of the proposed control scheme at controlling the shape the DLO. Thus, we used fixed parameters without online parameter adaptation. The objective was to investigate the positional error between the current and goal states of the DLO after manipulation with the proposed control scheme.

#### *Implementation and setting of the simulation*

The differentiable simulation of the DLO was implemented by using the automatic differentiation framework Taichi [51], which allows for parallel computation and reduces the computational time to obtain the analytic gradients. Adam was used for optimization. The rest of the proposed control scheme was implemented in Python. A computer with an Intel Xeon Gold 6242 CPU and Quadro RTX 6000 GPU was used for the simulations. The DLO was represented by 40 nodes ($n = 40$) and was grasped at its ends by end-effectors. As assumed in the problem setting presented in SubSect. "Problem setting", there is no self-intersection during the manipulation of the DLO. To prevent DLO from self-intersecting, we ignored the twist thereby limiting the DoFs of each action to only translations and without rotations ($\mu = 2 \times 3$). Table 1 presents the main simulation parameters related to the deformation of the DLO. The MPC hyperparameters were set to $T = 400, H = 5, \mathcal{I} = 30$, and $\omega_1 = 1.0\ \mathrm{m}^{-1}, \omega_2 = 0.01\ \mathrm{m}^2, \omega_3 = 0.5\ \mathrm{s}^2/\mathrm{m}^2$. The

**Table 1** Main simulation parameters of the DLO

| Parameters | Meaning | Value |
|---|---|---|
| $L$ | Length (m) | 0.45 |
| $m$ | Mass (kg) | 0.5 |
| $I$ | Inertia (kg · m$^2$) | 0.05 |
| $\Delta t$ | Time interval (sec) | 0.01 |
| $\xi_v$ | Damping coefficient of the linear velocity | 0.1 |
| $\xi_\omega$ | Damping coefficient of the angular velocity | 0.3 |
| $\beta_b$ | Stiffness coefficient of the bending (m$^2$/N) | 0.003 |

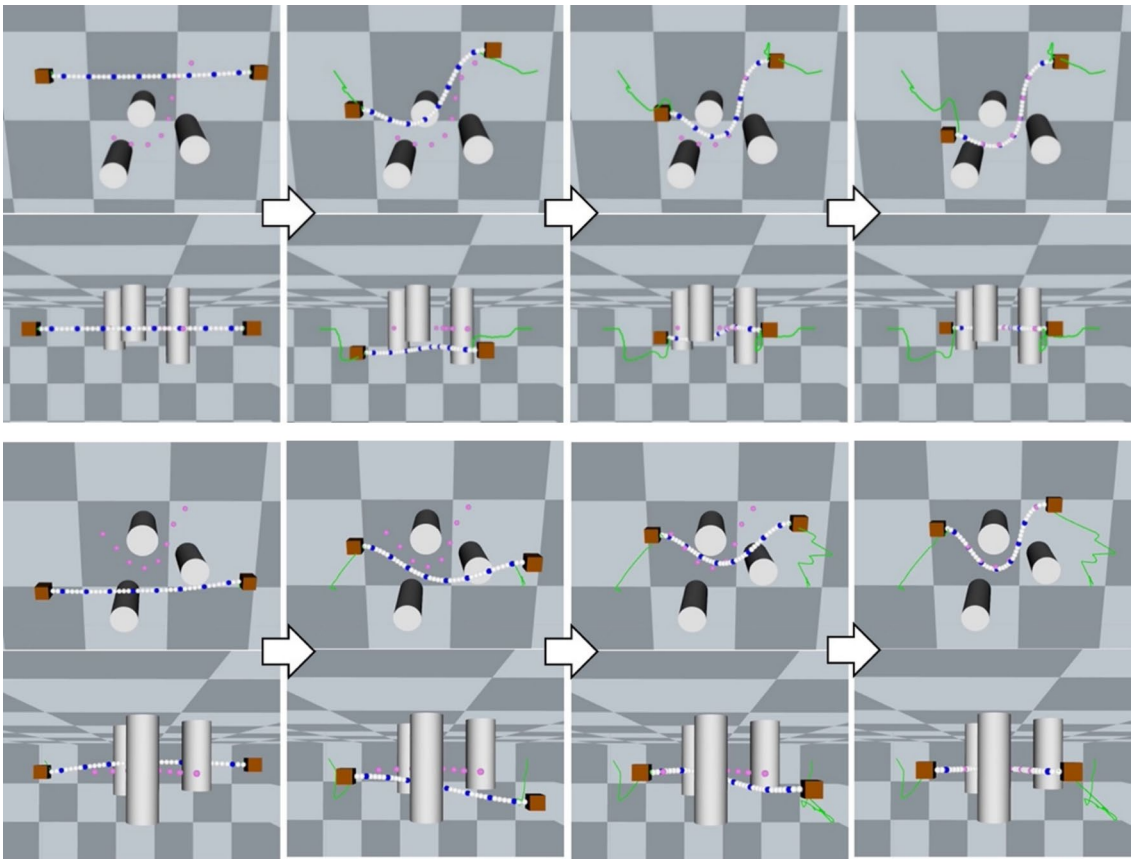


**Fig. 5** Snapshots of two simulations of DLO manipulation tasks. The top row shows the front view, and the bottom row shows the top view. The magenta points indicate the goal states, the blue points are the corresponding nodes on the DLO, and the green lines indicate the actions generated by the proposed control scheme

initial state of the DLO was a straight line (Fig. 5), and the goal state was randomly generated. Three cylinders with identical radii of 0.03 m but varying heights of 0.13, 0.16, and 0.19 m were installed as obstacles on the vertical plane.

#### *Simulation results*

We tested the method on 10 different goal shapes, and the error for all cases was below 5 mm with an average error of approximately 2.7 mm. Figure 5 shows two examples of manipulation processes, which demonstrate the ability of the proposed control scheme to avoid obstacles and move the DLO from the initial state to the goal state. No prior data were collected to learn the dynamics of the DLO.

### Real-world experiments

We conducted real-world experiments to investigate the accuracy of the proposed control scheme on different types of DLOs and the effectiveness of online parameter adaptation at narrowing the sim-to-real gap. We also

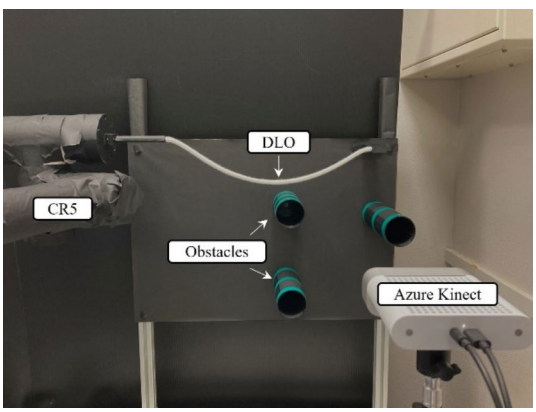



**Fig. 6** Setup for the real-world experiment

conducted an ablation study to evaluate the effectiveness of individual modules of the proposed control scheme.

#### *Instrument and setting*

As shown in Fig. 6, one end of the DLO was grasped by a CR5 robot while the other end remained fixed. To prevent the DLO from self-occlusion, which makes it difficult to measure the state of the DLO from the real world. Similar to the simulation setting, again we did not consider the twist thereby limiting the DoFs of the end-effector to only translations and without rotations ($\mu = 3$). The experiments were performed with a black background and white DLO. An Azure Kinect camera was used to acquire both RGB and depth images of the DLO, which were processed by using OpenCV. The positions of obstacles were determined by augmented reality markers. The MPC hyperparameters were identical to those used in the simulation experiments (Sect. "Simulations"). Two DLOs were tested using three different goal states. The first DLO was a softer rope with a length of 0.45 m and diameter of 8 mm. The second DLO was a harder cable with a length of 0.55 m and diameter of 5 mm. The states of the DLOs were represented by 30 and 40 nodes, respectively.

Figure 7 shows the markerless perception algorithm that we used to obtain the actual state of the DLO. The inputs consisted of RGB and depth images that contained the DLO, the length $L$ of the DLO, and the number n of nodes used to represent the state of the DLO. An RGB-D camera was used to capture the images from which the length was measured. The number of nodes was artificially defined to ensure adequate representation of the DLO geometry. Segmenting the DLO region in the image is an essential step that can be done by traditional image processing or data-driven approaches, but this was beyond the scope of this study. Therefore, we set the background to a pure black color that contrasted with the white color of the DLO so that the DLO could be segmented by binarization. Then, the centerline of the DLO was extracted by using skeletonization. The pixels of the centerline were projected into a 3D point cloud and sorted. The first and last points in the sorted point cloud were treated as the first node $\boldsymbol{x}_1$ and last node $\boldsymbol{x}_n$. The remaining nodes were then searched for in the point cloud by a greedy algorithm, which ensured that the error of the Euclidean distance between neighboring nodes was under a given threshold $\epsilon$ with respect to $L/(n-1)$. Once all the nodes were found, a chain was obtained consisting of nodes that were uniformly distributed over the DLO. This approach enabled quick detection of the actual state of the DLO while keeping it physically inextensible.

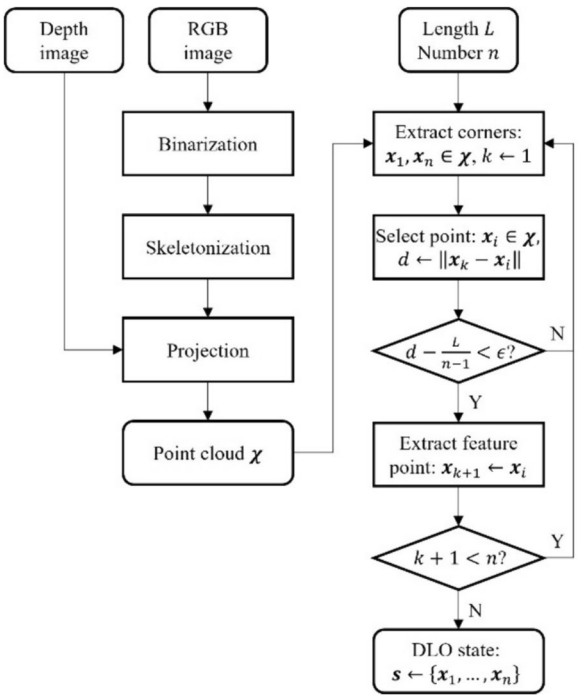



**Fig. 7** Markerless algorithm for obtaining the state of the DLO

#### *Real-world experimental results*

Table 2 presents the differences between the model parameters of the rope and cable owing to the differences in their properties. In particular, the cable had a larger stiffness coefficient than the rope, which is consistent with their material properties. Figure 8 shows the manipulation process of the DLOs. The visual perception algorithm allowed for robust real-time acquisition of the DLO state for online adaptation of model parameters. Even when the physical properties of the DLO were altered, the proposed control scheme could adapt the parameters to reduce the sim-to-real gap. Thus, we were able to successfully move the DLO to avoid obstacles and reach the goal state. As presented in Table 3, the average error in the real-world experiments was approximately 1 cm. Each error was defined as the average of the sum of the node-by-node Euclidean distances between the state after the manipulation $\boldsymbol{s}_T$ and the goal state $\boldsymbol{s}^g$, i.e.

**Table 2** Parameter adaptation

| Parameters | Initial value | Adapted value (rope) | Adapted value (cable) |
|---|---|---|---|
| $m$(kg) | 1.0 | 0.984 | 1.017 |
| $I$(kg · m$^2$) | 0.05 | 0.072 | 0.005 |
| $\xi_v$ | 0.1 | 0.006 | 0.021 |
| $\xi_\omega$ | 0.3 | 0.290 | 0.298 |
| $\beta_b$(m$^2$/N) | 0.003 | 0.0057 | 0.0062 |

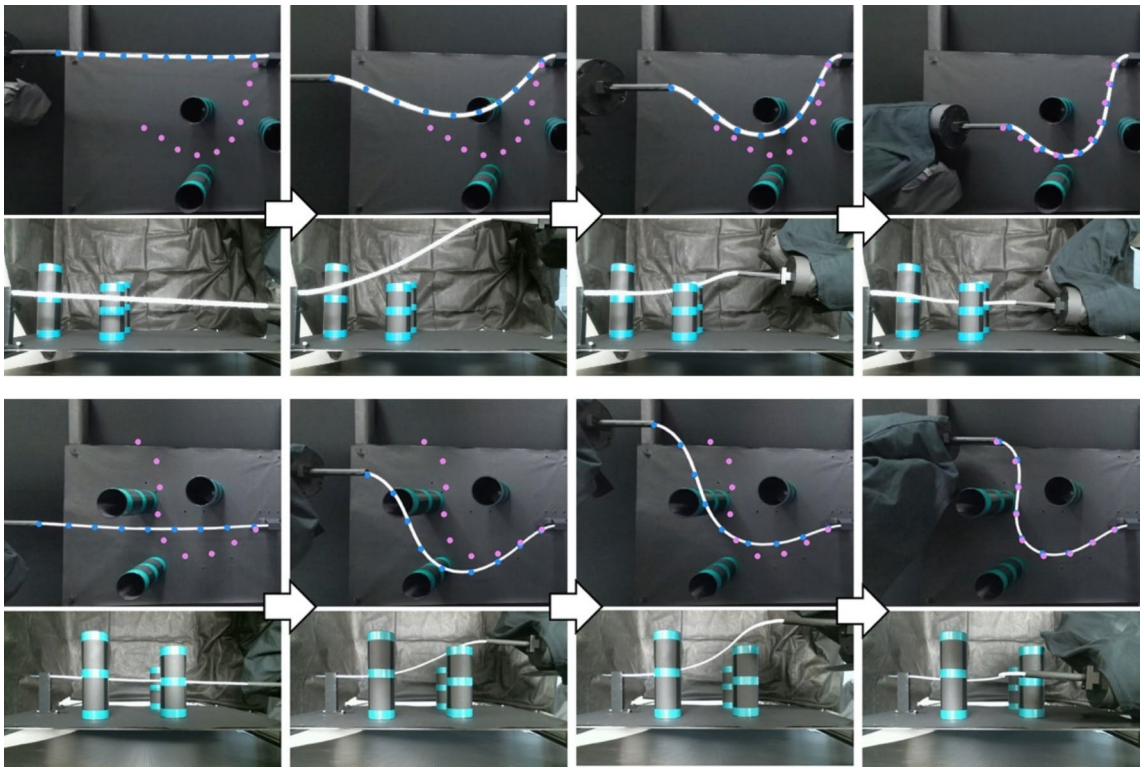


**Fig. 8** Snapshots of manipulation of the rope (top) and cable (below) in real-world experiments with different goal states and obstacles. The magenta points indicate the goal states, the blue points are the results of the markerless perception algorithm

**Table 3** Errors (in meters) of shape control in the real world

| | Case1 | Case2 | Case3 | Average |
|---|---|---|---|---|
| Rope | 0.018 | 0.011 | 0.009 | 0.0126 |
| Cable | 0.010 | 0.011 | 0.007 | 0.0093 |

$\|\boldsymbol{s}_T - \boldsymbol{s}^g\| = \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{x}_i^g\|$. Several factors contributed to the larger error than in the simulation, such as limitations in the camera accuracy, errors in robot–camera calibration, and imprecise DLO modeling. However, because we did not use any markers to acquire the state of the DLO and did not learn its dynamics based on any prior data, we believe that the proposed control scheme is effective at controlling the shape of the DLO. Movie 1 in the supplementary information demonstrate the manipulation of the DLOs in detail.

### *Results of the ablation study*

*Neutral networks* We conducted simulations to test the contribution of the NN to the shape control of the DLO. In scenarios without the NN, the DLO simulation constructed is end-to-end differentiable, allowing for the direct calculation of the gradient of the objective function value with respect to the action. Subsequently, a gradient descent method, such as Adam, was employed to optimize the sequence of actions for each MPC horizon. Fig. 9 (left) shows the results. For a given goal state and timespan, the average and variance of the control error were significantly reduced with the NN than without it. Despite our best efforts to optimize the action sequences by tuning the hyperparameters, the results still had a high probability of falling into a local optimum without the NN.

*Online parameter adaptation* Next, we investigated the contribution of online parameter adaptation to shape control using the fixed parameters given in Table I for the rope. The goal state was GOAL 1, and no online parameter adaptation was enabled. The results are shown in Fig. 9 (right). The error converged to a smaller value with online parameter adaptation than without, which can be attributed to narrowing of the sim-to-real gap. As shown in Fig. 10, a large discrepancy was observed between the simulated and real-world states without online parameter adaptation, which caused the shape control to converge to a local optimum that ultimately left the DLO in a position far from the goal state.

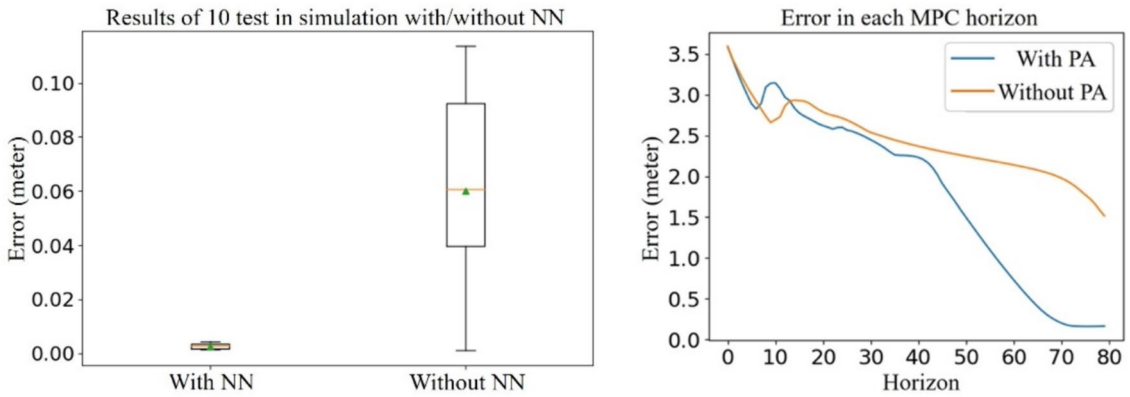



**Fig. 9** Ablation study of the NN and online parameter adaptation (PA). The left image shows the average and variance of the error in 10 goal states with and without the NN. The right image shows the update of the shape control error with and without PA for the goal state

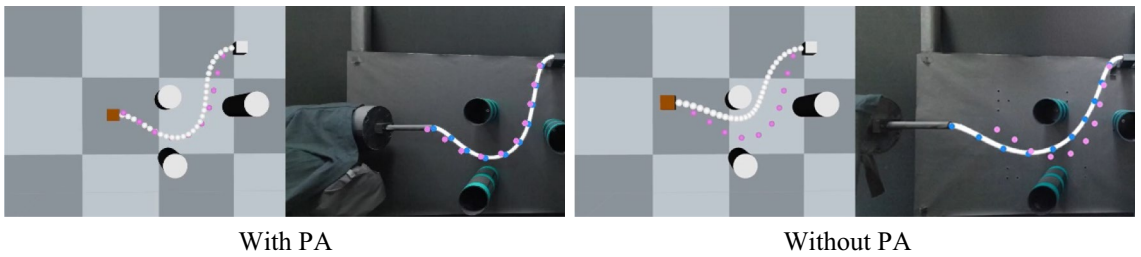



**Fig. 10** States of the DLO after manipulation in both the simulation and real world with online parameter adaptation (PA) (left) and without (right). With PA, the sim-to-real gap was small and had a minor influence on the real-world results. Without PA, the sim-to-real gap was large and resulted in a more significant control error

## Conclusion

In this paper, we proposed a control scheme for DLOs that eliminates the need for a priori data and specialized controllers. Our scheme uses differentiable simulation to facilitate not only forward dynamics transfer but also error backpropagation to obtain gradients useful for task optimization, and the shape control combines MPC with an embedded NN controller and online parameter adaptation to narrow the sim-to-real gap. Simulations and real-world experiments showed that the proposed control scheme was able to accurately manipulate DLOs into desired shapes while avoiding obstacles with average errors of 3 mm and 1 cm, respectively. Ablation studies were conducted that demonstrated the necessity of the NN controller and online parameter adaptation module for the proposed control scheme.

However, the proposed control scheme still has some limitations. First, the sim-to-real gap is still present, which is inevitable because DLOs are not made of completely isotropic materials in the real world, and they are prone to plastic deformation and perception errors. Therefore, building a perfect model that completely matches the real-world deformation of a DLO is not possible, which explains the much larger control error in the real-world experiment than in simulation. Second, our study was focused on control rather than planning, so the path of actions may not be globally optimal. In addition, the control actions were limited to translation because the DLO interacting with itself was not considered. Third, the computation time was relatively long at approximately 1.7 s to complete a control horizon. This limitation can be attributed to two primary factors. The first reason is that the more precise the simulation model is, the greater the number of computational steps required, which in turn results in a more complex computational graph for automatic differentiation. The

second reason is that the currently employed optimization method, namely gradient descent, necessitates a substantial number of iterations before the error converges. To reduce the computational time, some unnecessary computational steps may need to be removed from the computational graph of automatic differentiation. Furthermore, more efficient optimization algorithms would need to be employed to substitute the simple gradient descent method and thus decrease the number of optimization iterations. Finally, the existing control scheme is unable to effectively address the issue of shaping the DLO through contact with the external environment. This is because the constructed objective function solely considers the artificial potential field for obstacle avoidance, which would result in the DLO maintaining a distance from the obstacle. To achieve the desired outcome of contact with the external environment, it is necessary to modify the objective function accordingly.

In future work, we plan to investigate tracking algorithms to improve the robustness of the proposed control scheme against occlusion and to refine the model of the DLO dynamics to further narrow the sim-to-real gap. We also plan to achieve more complex control tasks, particularly in instances where contact with the external environment is necessary to shape the DLO. To this end, a novel objective function that can discern which contact is beneficial and which is detrimental may be required.

## Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s40648-024-00283-1.

Additional file 1.

**Acknowledgements**
This work was supported by JST SPRING, Grant Number JPMJSP2144 (Shinshu University).

**Author contributions**
All authors contributed to develop the method. C.Y conceived and designed this study, conducted experiments, and wrote the article. K.Y supervised this study and wrote the article. All authors read and approved the manuscript.

**Funding**
Not applicable.

**Availability of data and materials**
No datasets were generated or analysed during the current study.

## Declarations

**Competing interests**
The authors declare no competing interests.

Received: 30 April 2024 Accepted: 20 September 2024
Published online: 27 September 2024

**References**

1. Yin H, Varava A, Kragic D (2021) Modeling, learning, perception, and control methods for deformable object manipulation. Sci Robot. https://doi.org/10.1126/scirobotics.abd8803
2. Arriola-Rios VE, Guler P, Ficuciello F et al (2020) Modeling of deformable objects for robotic manipulation: a tutorial and review. Front Robot AI 7:82. https://doi.org/10.3389/frobt.2020.00082
3. Laezza R, Gieselmann R, Pokorny FT, Karayiannidis Y (2021) ReForm: a robot learning sandbox for deformable linear object manipulation. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE: Xi'an, China, pp 4717–4723
4. Wang W, Balkcom D (2018) Knot grasping, folding, and re-grasping. Int J Robot Res 37:378–399. https://doi.org/10.1177/0278364918754676
5. Chen K, Bing Z, Wu F, et al (2023) Contact-aware shaping and maintenance of deformable linear objects with fixtures. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp 1–8
6. Zhang Z, Dequidt J, Back J et al (2019) Motion control of cable-driven continuum catheter robot through contacts. IEEE Robot Autom Lett 4:1852–1859. https://doi.org/10.1109/LRA.2019.2898047
7. Ying C, Yamazaki K (2023) Motion generation for shaping deformable linear objects with contact avoidance using differentiable simulation *. In: 2023 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, Koh Samui, Thailand, pp 1–8
8. Lv N, Liu J, Xia H et al (2020) A review of techniques for modeling flexible cables. Comput Aided Des 122:102826. https://doi.org/10.1016/j.cad.2020.102826
9. Lv N, Liu J, Ding X et al (2017) Physically based real-time interactive assembly simulation of cable harness. J Manuf Syst 43:385–399. https://doi.org/10.1016/j.jmsy.2017.02.001
10. Koessler A, Filella NR, Bouzgarrou BC, et al (2021) 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). pp 1637–1643
11. Xu L, Liu Q (2018) Real-time inextensible surgical thread simulation. Int J CARS 13:1019–1035. https://doi.org/10.1007/s11548-018-1739-1
12. Yang Y, Stork JA, Stoyanov T (2022) Learning differentiable dynamics models for shape control of deformable linear objects. Robot Auton Syst 158:104258. https://doi.org/10.1016/j.robot.2022.104258
13. Wang C, Zhang Y, Zhang X et al (2022) Offline-online learning of deformation model for cable manipulation with graph neural networks. IEEE Robot Autom Lett 7:5544–5551. https://doi.org/10.1109/LRA.2022.3158376
14. Yu M, Lv K, Zhong H et al (2023) Global model learning for large deformation control of elastic deformable linear objects: an efficient and adaptive approach. IEEE Trans Robot 39:417–436. https://doi.org/10.1109/TRO.2022.3200546
15. McConachie D, Power T, Mitrano P, Berenson D (2020) Learning when to trust a dynamics model for planning in reduced state spaces. IEEE Robot Autom Lett 5:3540–3547. https://doi.org/10.1109/LRA.2020.2972858
16. Huang Y, Xia C, Wang X, Liang B (2023) Learning graph dynamics with external contact for deformable linear objects shape control. IEEE Robot Autom Lett 8:3891–3898. https://doi.org/10.1109/LRA.2023.3264764
17. Duenser S, Bern JM, Poranne R, Coros S (2018) Interactive robotic manipulation of elastic objects. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp 3476–3481
18. Lv N, Liu J, Jia Y (2022) Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations. IEEE Trans Robot 38:2341–2353. https://doi.org/10.1109/TRO.2021.3139838
19. Petit A, Ficuciello F, Fontanelli GA, et al (2017) Using physical modeling and rgb-d registration for contact force sensing on deformable objects: In: Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics. SCITEPRESS - Science and Technology Publications, Madrid, Spain.
20. Güler P, Pieropan A, Ishikawa M, Kragic D (2017) Estimating deformability of objects using meshless shape matching. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).
21. Liu F, Su E, Lu J et al (2023) Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics. IEEE Robot Autom Lett 8:3964–3971. https://doi.org/10.1109/LRA.2023.3264766

22. Nozaki K, Ying C, Matsuura Y, Yamazaki K (2023) Manipulation planning for wiring connector-attached cables considering linear object's deformability. Int J Autom Technol. https://doi.org/10.20965/ijat.2023.p0399
23. S L (1998) Rapidly-exploring random trees : a new tool for path planning. Research Report 9811
24. Amato NM, Wu Y (1996) A randomized roadmap method for path and manipulation planning. In: Proceedings of IEEE International Conference on Robotics and Automation.
25. Bretl T, McCarthy Z (2014) Quasi-static manipulation of a Kirchhoff elastic rod based on a geometric analysis of equilibrium configurations. Int J Robot Res 33:48–68. https://doi.org/10.1177/0278364912473169
26. Moll M, Kavraki LE (2006) Path planning for deformable linear objects. IEEE Trans Rob 22:625–636. https://doi.org/10.1109/TRO.2006.878933
27. Mishani I, Sintov A (2022) Real-time non-visual shape estimation and robotic dual-arm manipulation control of an elastic wire. IEEE Robot Autom Lett 7:422–429. https://doi.org/10.1109/LRA.2021.3128707
28. Roussel O, Fernbach P, Taïx M (2020) Motion planning for an elastic rod using contacts. IEEE Trans Autom Sci Eng 17:670–683. https://doi.org/10.1109/TASE.2019.2941046
29. Li Y, Wu J, Zhu J-Y, et al (2019) Propagation networks for model-based control under partial observation. In: 2019 International Conference on Robotics and Automation (ICRA). IEEE, Montreal, QC, Canada.
30. Yan M, Zhu Y, Jin N, Bohg J (2020) Self-supervised learning of state estimation for manipulating deformable linear objects. IEEE Robot Autom Lett 5:2372–2379. https://doi.org/10.1109/LRA.2020.2969931
31. Zhang W, Schmeckpeper K, Chaudhari P, Daniilidis K (2021) Deformable linear object prediction using locally linear latent dynamics. In: 2021 IEEE International Conference on Robotics and Automation (ICRA).
32. Yan W, Vangipuram A, Abbeel P, Pinto L (2021) Learning predictive representations for deformable objects using contrastive estimation. In: Proceedings of the 2020 Conference on Robot Learning. PMLR.
33. Zhu J, Navarro-Alarcon D, Passama R, Cherubini A (2021) Vision-based manipulation of deformable and rigid objects using subspace projections of 2D contours. Robot Auton Syst 142:103798. https://doi.org/10.1016/j.robot.2021.103798
34. Lagneau R, Krupa A, Marchal M (2020) Automatic shape control of deformable wires based on model-free visual servoing. IEEE Robot Autom Lett 5:5252–5259. https://doi.org/10.1109/LRA.2020.3007114
35. Yang B, Lu B, Chen W et al (2023) Model-free 3-D shape control of deformable objects using novel features based on modal analysis. IEEE Trans Rob 39:3134–3153. https://doi.org/10.1109/TRO.2023.3269347
36. Berenson D (2013) Manipulation of deformable objects without modeling and simulating deformation in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE: Tokyo.
37. Zhu J, Navarro B, Fraisse P, et al (2018) Dual-arm robotic manipulation of flexible cables In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE: Madrid.
38. Lin X, Wang Y, Olkin J, Held D (2021) SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation. In: Proceedings of the 2020 Conference on Robot Learning. PMLR
39. Han H, Paul G, Matsubara T (2017) Model-based reinforcement learning approach for deformable linear object manipulation In: 2017 13th IEEE Conference on Automation Science and Engineering (CASE). IEEE: Xi'an.
40. Wu Y, Yan W, Kurutach T, et al (2020) Learning to manipulate deformable objects without demonstrations. In: Robotics: Science and Systems XVI. Robotics: Science and Systems Foundation
41. Hiruma H, Ito H, Mori H, Ogata T (2022) Deep active visual attention for real-time robot motion generation: emergence of tool-body assimilation and adaptive tool-use. IEEE Robot Autom Lett 7:8550–8557. https://doi.org/10.1109/LRA.2022.3187614
42. Hayashi K, Sakaino S, Tsuji T (2022) An independently learnable hierarchical model for bilateral control-based imitation learning applications. IEEE Access 10:32766–32781. https://doi.org/10.1109/ACCESS.2022.3155255
43. Liang J, Lin M, Koltun V (2019) Differentiable cloth simulation for inverse problems. In: Advances in Neural Information Processing Systems. Curran Associates, Inc.
44. Degrave J, Hermans M, Dambre J, Wyffels F (2019) A differentiable physics engine for deep learning in robotics. Front Neurorobot 13:6. https://doi.org/10.3389/fnbot.2019.00006
45. Chen S, Werling K, Wu A, Liu CK (2023) Real-time model predictive control and system identification using differentiable simulation. IEEE Robot Autom Lett 8:312–319. https://doi.org/10.1109/LRA.2022.3226027
46. Chen S, Liu Y, Yao SW et al (2022) DiffSRL: learning dynamical state representation for deformable object manipulation with differentiable simulation. IEEE Robot Autom Lett 7:9533–9540. https://doi.org/10.1109/LRA.2022.3192209
47. Millard D, Preiss JA, Barbič J, Sukhatme GS (2023) Parameter estimation for deformable objects in robotic manipulation tasks. In: Billard A, Asfour T, Khatib O (eds) Robotics Research. Springer Nature Switzerland, Cham, pp 239–251
48. Yang Y, Stork JA, Stoyanov T (2022) Online model learning for shape control of deformable linear objects. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, Kyoto, Japan, pp 4056–4062
49. Kugelstadt T, Schömer E (2016) Position and orientation based Cosserat rods. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association, Goslar, DEU, pp 169–178
50. Macklin M, Müller M, Chentanez N (2016) XPBD: position-based simulation of compliant constrained dynamics in proceedings of the 9th international conference on motion in games. ACM: Burlingame California.
51. Hu Y, Anderson L, Li T-M, et al (2020) Difftaichi: differentiable programming for physical simulation. arXiv preprint arXiv


## Publisher's Note