Online Model Learning for Shape Control of Deformable Linear Objects

Yuxuan Yang, Johannes A. Stork, and Todor Stoyanov

***Abstract*— Traditional approaches to manipulating the state of deformable linear objects (DLOs) — i.e., cables, ropes — rely on model-based planning. However, constructing an accurate dynamic model of a DLO is challenging due to the complexity of interactions and a high number of degrees of freedom. This renders the task of achieving a desired DLO shape particularly difficult and motivates the use of model-free alternatives, which while maintaining generality suffer from a high sample complexity. In this paper, we bridge the gap between these fundamentally different approaches and propose a framework that learns dynamic models of DLOs through trial-and-error interaction. Akin to model-based reinforcement learning (RL), we interleave learning and exploration to solve a 3D shape control task for a DLO. Our approach requires only a fraction of the interaction samples of the current state-of-the-art model-free RL alternatives to achieve superior shape control performance. Unlike offline model learning, our approach does not require expert knowledge for data collection, retains the ability to explore, and automatically selects relevant experience.**

I. Introduction

Deformable linear objects (DLOs), such as cables, ropes, hoses, sutures, etc., are widely used in industry and daily life settings. This has led to a growing interest in deformable linear object manipulation in the field of robotics [1]–[3], like inserting and threading a needle [4], [5], surgical suturing [6], [7], or controlling the shape of a rope [8]. Despite these recent studies, deformable objects still pose a significant challenge [2] compared to the better understood problems arising in traditional rigid object manipulation. DLOs in particular are a category of deformable objects with complex dynamics in 3D space due to their twisting and bending behavior. They can move in all three directions in space and deform by bending and twisting due to external forces. Modeling the dynamics of DLOs is still an open question in robotics [2]. Consequently, this leads to difficulties in applying classical model-based control methods. These methods assume precise DLO dynamics or kinematics models based on expert knowledge in physics [9] or topology [7]. The performance of these methods strongly relies on the accuracy and efficiency of the proposed DLO models.

In contrast, learning-based methods are a promising alternative. Instead of designing an analytical model for DLOs' dynamics, we can learn a data-driven model directly by observing a manipulated DLO [10]–[13]. Data-driven models based on neural networks are differentiable by default, which makes them a good candidate for model predictive control (MPC) for manipulation tasks. One limitation is that a large quantity of annotated data is needed for training, which is not practical in many real-world applications. Data collection experiments also need a proper design so the collected data contains valuable information for learning. An effective and efficient data collection experiment can be hard to design and requires expert knowledge. Alternatively, we can learn a manipulation policy directly by reinforcement learning (RL) through a trial-and-error learning process. Despite many recent advances in RL [14], model-free RL methods usually require millions of interactions with the environment, which can be impractical [15]. In contrast, model-based RL methods generally have better sample-efficiency [16], but face challenges when generalizing to high-dimensional state spaces such as those occurring in the DLO shape control tasks.

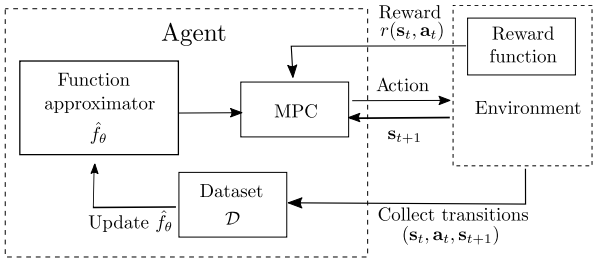

Fig. 1. Illustration of our online model learning framework with model predictive control.

Previously, we demonstrated how to learn a DLO model offline using pre-recorded data and how to use the learned model in a model predictive controller (MPC) to achieve the desired shapes [13]. A large amount of required data and the data collection process design make related real-world applications cumbersome. To address this issue, we present an improved framework for sample-efficient online dynamics model learning and shape control of a DLO through trial-and-error interactions. Our framework is similar to the idea of model-based RL, interleaving learning and exploration in a shape control task, as shown in Fig. 1. In particular, we continue using a neural network structure and state representation proposed in our previous work [13] as the model to approximate the dynamics of a DLO.

To demonstrate the effectiveness of the proposed framework, we train and evaluate our method in simulation using AGX Dynamics[1], a commercial simulation physics engine with specialized models for DLOs [17], [18]. We compare our framework to a model-based RL algorithm PILCO [19] and two model-free ones, Twin Delayed Deep Deterministic

All authors are with the Autonomous Mobile Manipulation Lab, Center for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden. This work was partially supported by Vinnova / SIP-STRIM projects 2019-05175, and was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

[1]https://www.algoryx.se/agx-dynamics/

Policy Gradient (TD3) [20] and Soft Actor-Critic (SAC) [21]. TD3 trains a deterministic policy while SAC trains a stochastic policy. We demonstrate that: (1) Our approach achieves better sample efficiency than RL algorithms on both specific and general tasks, and (2) compared to the offline method, our approach is more sample efficient when solving a specific shape control task without the requirement of expert knowledge for designing the data collection process.

II. Related Work

A. Models for Manipulating DLOs

Modeling deformable linear object dynamics is still an active research topic. Traditionally, analytical physics-based models are used to simulate DLOs. Generally, models based on finite element methods are used when high physical accuracy is desired. If lower accuracy is permissible, models based on spring-mass systems are both simpler and less computationally expensive [2]. In computer graphics, position-based dynamics are widely used because they are efficient and visually-plausible [22], [23]. However, both the faster methods that trade physical accuracy for tractability and physically accurate methods with high computational complexity are ill-suited for control tasks.

Another solution to DLO manipulation is servo control, which only calculates a local deformation model instead of a full dynamics model. Zhu et al. [24] formulate a velocity control law by using an online estimated local deformation model. Their method does not guarantee asymptotic control stability because of under-actuation. Navarro-Alarcon et al. [25], [26] continuously update the deformation Jacobian matrix of a deformable object in real time by visual feedback. In contrast, Lagneau et al. [27] iteratively update a Jacobian matrix using weighted least-squares minimization with a sliding window to achieve DLO manipulating by an adaptive model-free method. These methods require a Jacobian matrix to establish a mapping between a shape description and control inputs, which requires scenario-depended expert knowledge. They also tend to be stuck in local minima while, in contrast, our approach can get out of local minima due to the learned dynamics model and MPC.

B. Data-Driven Models for DLO dynamics

Neural networks are an alternative to model object dynamics [28]. Interaction networks (INs) combine neural networks and compositional modeling to learn complex dynamics from observed data [10]. In the context of DLOs, INs utilize a graph network [29] to learn pair-wise local interactions between different segments of a DLO. An issue in deploying INs to simulate DLOs is propagating the effects of interaction across object segments. This is because local information is insufficient for modeling dynamics while effects among distant segments are important but remain challenging to predict [11]. Different means of interaction propagation [12], [30], [31] have been introduced to address this issue. Nevertheless, most of these methods are only verified on DLOs in 2D space in simulation.

In our previous work [13], we integrate a recurrent model into an IN model to leverage both compositional models of DLO dynamics from INs and learning to propagate interaction from the recurrent model. The forward prediction performance of the proposed model is tested on DLOs in 3D space both in simulation and in the real world. We also implement a model predictive controller to accomplish a DLO shape control task in simulation based on the learned data-driven model. However, to train an effective model, a well-designed data collection process and a large number of labeled data are needed, which poses challenges in real-world applications. To address this issue, in this paper, we achieve a sample-efficient online model learning for a DLO shape control task through trial and error.

C. Reinforcement Learning for Manipulating DLOs

Recently, deep reinforcement learning algorithms, like temporal difference learning [32], actor-critic methods [33], [34], and policy gradients [15] have successfully been applied to complex policy learning in high-dimensional state spaces. However, it is difficult to use these model-free reinforcement learning (RL) algorithms in real-world scenarios because of the high demand for interaction samples. Compared to model-free counterparts, model-based RL algorithms usually have better sample efficiency [16]. Thus, model-based RL algorithms perform better in some robot control tasks both in simulations and in real-world scenarios [19], [35]. However, these algorithms employ function approximators such as Gaussian processes [19] and Gaussian mixture models [36]. PILCO [19], for instance, is a model-based policy search method that approximates system dynamics by a Gaussian process. These function approximators have limitations and do not perform as well when the state space is high-dimensional. One similar work uses PILCO to manipulate a DLO in 2-D space [37], while in our case, manipulating a DLO in 3D space poses additional challenges due to higher dimensions in both state and action space and more complex nonlinear dynamics. Attempts have been made to improve PILCO by replacing its parameterized policy with model predictive control (MPC) [38] and by replacing its Gaussian process module with neural network models which feed into an MPC scheme [39]. Both of these approaches have demonstrated better sample efficiency, but only in simple tasks with a relatively low-dimensional state space such as double pendulum and quadrupedal robot control. In this paper, we take a similar approach but use a specialized neural network model we proposed in our previous work [13] which has the capacity to approximate DLOs' dynamics.

III. Background

Reinforcement learning (RL) learns a policy that maximizes the sum of future rewards. An RL problem is usually formalized as a Markov Decision Process. In a deterministic system, at a time step t, an agent takes a certain action $\boldsymbol{a}_t \in \mathcal{A}$ at state $\boldsymbol{s}_t \in \mathcal{S}$. Then it transitions to a new state $\boldsymbol{s}_{t+1}$ with a reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The transition follows

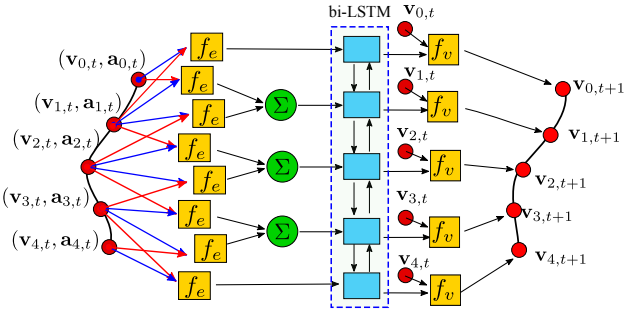

Fig. 2. Neural network structure for modeling a DLO. The vertices $\boldsymbol{v}_{i,t}$ correspond intersections between neighbor Cosserat rod segments. $\boldsymbol{a}_{i,t}$ represents the action applied on the vertices. f_e is the relation-centric network and f_v is the object-centric network.

a dynamics function, $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, which is usually unknown. The agent with optimal policy will take actions that maximize the discounted sum of future rewards, e.g. $\sum_{t'=t}^{\infty} \gamma^{t'-t}\, r(\boldsymbol{s}_{t'}, \boldsymbol{a}_{t'})$, where $\gamma \in (0, 1]$ is a discount factor. In this work, we use a simple reward function for the DLO shape control task, see Section V-B.

Model-based RL uses a dynamics model for prediction. The dynamics model of the system is $\hat{\boldsymbol{s}}_{t+1} = \hat{f}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)$, where $\hat{f}_\theta$ is a dynamics function approximator which can be parameterized models (neural networks) and non-parametric models (Gaussian processes). Actions in a finite-horizon discounted Markov decision process can be optimizated by solving

$$\mathcal{A}_t = (\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+H}) \tag{1}$$

$$= \underset{\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+H}}{\arg\max} \sum_{t'=t}^{t+H} \gamma^{t'-t}\, r(\hat{\boldsymbol{s}}_{t'}, \boldsymbol{a}_{t'}) \tag{2}$$

In this paper, we use model predictive control to compensate for modeling errors. Therefore, after solving this optimization problem at a time step, only the first action $\boldsymbol{a}_t$ in the sequence will be executed. In the next time step, the optimization problem is then solved again with the latest state.

IV. Online Model Learning For DLO Manipulation

This section presents our online model learning (RL) approach for DLO manipulation. In Section IV-A, we briefly introduce the data-driven model for learning DLO dynamics. Then, in Section IV-B, we explain the model predictive control used in the paper. Last, in Section IV-C, we summarize our online model learning framework.

A. Data-driven Model for DLO Dynamics

We describe a DLO as a series of connected segments based on an explicitly discretized Cosserat rod [6]. We use a neural network structure proposed in our previous work [13] with small modifications as a function approximator $\hat{f}_\theta$ in the online model learning framework. The function approximator can predict the future state, $\hat{\boldsymbol{s}}_{t+1}$, given current state $\boldsymbol{s}_t$ and current action $\boldsymbol{a}_t$, i.e. $\hat{\boldsymbol{s}}_{t+1} = \hat{f}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Here, $\boldsymbol{s}_t$ contains positions and orientations of the segments in a DLO.

Our data-driven model of DLO dynamics is based on the Interaction Network (IN) method [10]. The IN method assumes that the overall dynamics of the particle-based system are made up of local interactions between related particles. Therefore, it is a generic dynamics model for particle-based systems represented by a directed graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. As shown in Fig. 2, the graph for a DLO is a chain with vertices representing different segments of the DLO connected by edges. The vertices, $\boldsymbol{v} \in \mathcal{V}$, represent the particles. The edges $\boldsymbol{e} \in \mathcal{E}$ represent their relations encoded by $\boldsymbol{e}_{ij} = \eta\,(\boldsymbol{v}_i, \boldsymbol{a}_i, \boldsymbol{v}_j, \boldsymbol{a}_j)$. $\boldsymbol{a}_i$ is the external force applied on vertex i. A simple formulation of η is directly concatenating $\boldsymbol{v}_i$, $\boldsymbol{a}_i$, $\boldsymbol{v}_j$, and $\boldsymbol{a}_j$. A directed edge, $\boldsymbol{e}_{ij}$, represents a relation where $\boldsymbol{v}_i$ is the receiver and $\boldsymbol{v}_j$ the sender. Therefore the dynamics model $\hat{f}_\theta$ based on the IN method maps a state

$$\boldsymbol{s}_t = (\{\boldsymbol{v}_{i,t} | \boldsymbol{v}_i \in V\}, \{\boldsymbol{e}_{ij,t} | \boldsymbol{e}_{ij} \in E\}) \tag{3}$$

with particle encodings $\boldsymbol{v}_{i,t}$ and relation encodings $\boldsymbol{e}_{ij,t}$ at time step t to the next state $\boldsymbol{s}_{t+1}$ with particle encodings $\boldsymbol{v}_{i,t+1}$ and relation encodings $\boldsymbol{e}_{ij,t+1}$. This mapping can be broken down into two components f_e and f_v. As seen in Eq. (4), the mapping f_e maps relation encodings to latent feature space. A vertex can exist in different relations as receivers, therefore we aggregate the features which belong to the same receiver. In Eq. (5), f_v maps aggregated features, $\boldsymbol{k}_i$, and particle encodings to predict future particle encodings.

$$\boldsymbol{k}_i = \sum_{\boldsymbol{e}_{ij} \in E} f_e(\boldsymbol{e}_{ij,t-1}) \tag{4}$$

$$\hat{\boldsymbol{v}}_{i,t} = f_v(\boldsymbol{v}_{i,t-1}, \boldsymbol{k}_i) \tag{5}$$

The mappings f_e and f_v are referred to as *relation-centric network* and *object-centric network*, respectively, in [10].

For now, only local interactions are taken into account at each time step while longer-reaching interactions which occur in DLOs are difficult to learn. To address this issue, a bidirectional long short-term memory (bi-LSTM) [40] module is introduced. LSTM perfectly matches the linear DLO structure when used spatially. We employ an LSTM cell recurrently along the DLO based on the assumption that all vertices in a DLO share identical dynamics. The LSTM learns how the effects propagate along the DLO from data without the requirement of setting and adjusting a hyperparameter related to iteration times as in [12], [31]. In position-based dynamics of DLOs, the implementation of a bilateral interleaving order to update the DLO state improve the stability [22], [23]. Similarly, we implement a bi-directional LSTM to simultaneously propagate the effects in both directions.

The structure of the model is shown in Fig. 2. We propagate the aggregated features $\mathbf{k}_i$ of each vertex to all the other vertices using an m-layer bi-directional LSTM, f_{LSTM}, along with two directions.

$$(\mathbf{h}_i^l, \mathbf{c}_i^l) = f_{LSTM}(\mathbf{k}_i, \mathbf{h}_{i-1}^l, \mathbf{c}_{i-1}^l) \tag{6}$$

$$(\mathbf{h}_i^r, \mathbf{c}_i^r) = f_{LSTM}(\mathbf{k}_i, \mathbf{h}_{i+1}^r, \mathbf{c}_{i+1}^r) \tag{7}$$

where $(\mathbf{h}_0^l, \mathbf{c}_0^l) = f_{LSTM}(\mathbf{k}_0, \mathbf{0}, \mathbf{0})$ and $(\mathbf{h}_N^r, \mathbf{c}_N^r) = f_{LSTM}(\mathbf{k}_N, \mathbf{0}, \mathbf{0})$, $\mathbf{h}_i$. The $\mathbf{c}_i$ are the hidden state vectors and the cell state vector in the LSTM with superscripts l and r indicate the direction of propagation.

We design propagated features by directly concatenating the last-layer hidden state vectors $\mathbf{h}_i^{l,m}$ and $\mathbf{h}_i^{r,m}$ because it is proven to be a better option than aggregated features [11]. The features which can contain effect information from all vertices in the chain then serve as input to f_v in Eq. (5).

For a more detailed explanation of the data-driven model, we refer to our previous work [13].

B. Model Predictive Control

Given a model of DLO dynamics as described in the previous section, we can rollout, that is iteratively predict the future states of a DLO, using the learned dynamics and optimize the control inputs $\boldsymbol{A}_t$ by maximizing the sum of future rewards in a time horizon.

Similar to Li et al. [30], we implement model predictive control (MPC) with a shooting method for DLO shape control tasks. Given a state $\boldsymbol{s}_t$ at time step t, a goal state, s_g, a data-driven dynamics model $\hat{f}_\theta$, and a sequence of control commands $\boldsymbol{A}_t = (\boldsymbol{a}_t, ..., \boldsymbol{a}_{t+H})$ for a time horizon H, we can calculate a sequence of predicted states as $\hat{\boldsymbol{S}} = (\hat{\boldsymbol{s}}_{t+1}, \ldots, \hat{\boldsymbol{s}}_{t+H})$ through forward simulation, $\hat{\boldsymbol{s}}_{t+1} = \hat{f}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Then, we can backpropagate gradients with respect to the control commands $\boldsymbol{A}_t$ from the sum of future rewards, $\sum_{t'=t}^{t+H-1} r(\hat{\boldsymbol{s}}_{t'}, \boldsymbol{s}_g)$, therefore, control commands can be optimized via gradient-based optimization methods.

In this paper, the reward function is the negative of $L(\hat{\boldsymbol{s}}_{t'}, \boldsymbol{s}_g)$. For simplicity, we base the loss function L on the Euclidean distance between the predicted positions of the segments in a DLO and the corresponding goal positions. Note that the loss function directly relates to the expected reward in (8) which we will formulate in the shape control task. We then use stochastic gradient descent (SGD) to optimize the control commands.

$$r(\hat{\boldsymbol{s}}_{t'}, \boldsymbol{s}_g) = -L(\hat{\boldsymbol{s}}_{t'}, \boldsymbol{s}_g) \tag{8}$$

$$\boldsymbol{A}_t = \arg\max_{\boldsymbol{A}_t} \sum_{t'=t}^{t+H-1} r(\hat{\boldsymbol{s}}_{t'}, \boldsymbol{s}_g) \tag{9}$$

$$\texttt{s.t. } \hat{\boldsymbol{s}}_{t'+1} = \hat{f}_\theta(\hat{\boldsymbol{s}}_{t'}, \boldsymbol{a}_{t'}),\ \hat{\boldsymbol{s}}_t = \boldsymbol{s}_t \tag{10}$$

This combination of a data-driven dynamics model with MPC leads to an advantage. Compared to model-free RL algorithms, once the model is trained, we can also accomplish a variety of similar tasks by simply changing the loss function, L, without the need for task-specific retraining.

C. Online Model Learning with MPC

The overall structure of our approach for online model learning with MPC is shown in Algorithm 1. It consists of initialization, updating model parameters, and on-policy data aggregation. We first collect initial training data by sampling a starting configuration and executing random actions at each time step during a time horizon T. Then we repeat the process several times to collect data from multiple trajectories. Training data are $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ as input and $\boldsymbol{s}_{t+1}$ as corresponding output labels. To increase the robustness of the trained data-driven model, we also add Gaussian noise to the input data during model learning.

Algorithm 1: Online Model Learning with MPC

Input : Max iteration times K
Trajectory time steps T
Current control inputs $\boldsymbol{A}$
The number of MPC trials per iteration N
Action noise distribution $\boldsymbol{A}_{\text{noise}}$

Collect dataset $\mathcal{D}$ of random trajectories
Initialize $\hat{f}_\theta$
for $k \leftarrow 1$ **to** K **do**
  update $\hat{f}_\theta$ using dataset $\mathcal{D}$
  for $n \leftarrow 1$ **to** N **do**
    Sample a random starting configuration $\boldsymbol{s}_1$
    for $t \leftarrow 1$ **to** T **do**
      acquire agent's current state $\boldsymbol{s}_t$
      execute MPC with shooting methods based on $\hat{f}_\theta$ to optimize action sequence $(\boldsymbol{a}_t, ..., \boldsymbol{a}_{t+H})$ (Equation. 8, 9, 10)
      With 0.2 probabability: $\boldsymbol{a}_t \leftarrow \boldsymbol{a}_t + \boldsymbol{a}_{\text{noise}}, \boldsymbol{a}_{\text{noise}} \sim \boldsymbol{A}_{\text{noise}}$
      execute action $\boldsymbol{a}_t$
      get new state $\boldsymbol{s}_{t+1}$
      add $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$ to $\mathcal{D}$
    end
  end
end

In the next state, we sample several random starting configurations for agents and control them using MPC trying to finish an assigned task. In the meantime, we collect these on-policy data and add them to the dataset $\mathcal{D}$. The initial dataset, which is generated randomly, is likely to be different from the trajectories an agent will finally execute to finish a task. Note that we add noise to the action with a probability of 0.2, which strengthens the exploration. We mitigate this discrepancy between the data state-action distribution and the model-based controller's distribution by collecting on-policy data and interleaving learning and exploration.

The online model learning framework continues switching between updating the data-driven dynamics model $\hat{f}_\theta$ and collecting newly generated data by model-based control until a predefined maximum iteration is reached.

V. Evaluation

A. DLO Simulation

Our ultimate goal is to implement online model learning with model predictive control (MPC) to achieve shape control of deformable linear objects (DLOs) in the real world. However, considering practical problems like DLO state estimation, we validate our methods in simulation. Beginning with simulation studies also makes comparisons to baseline

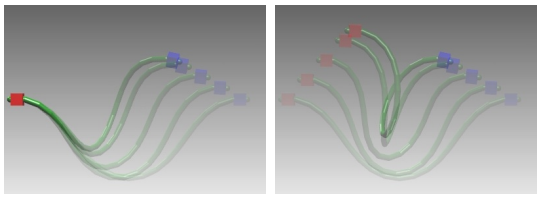

Fig. 3. The DLOs (in green) are approaching a desired target shape by operating on only one (left image) or both grippers (right image). One end of the DLO is controlled by a gripper (red) on the left and the other end is controlled by a gripper (blue) on the right.

model-free RL methods more practical since these generally require a very large number of samples, which is challenging for real-world experiments.

We build our simulation environment in Reform [41], which is a new sandbox building on AGX Dynamics for robot learning specializing in DLO manipulation. We opt for Reform because, firstly, AGX Dynamics provides a Cable module that specializes in DLOs with physically motivated physics properties like Young's modulus and Poisson's ratio. Secondly, it is because environments in Reform are based on the interfaces of OpenAI Gym [42], which is convenient for testing state-of-the-art RL algorithms.

The DLO used in the simulation is a lumped element model with a length of one meter, discretized into 20 segments. Young's modulus is set to 1×10^8 $\mathtt{N/m}^2$ in tension and 1×10^7 $\mathtt{N/m}^2$ in bending and twisting, and Poisson's ratio is set to 0.33. The two ends of a DLO are attached to two grippers which can both translate and rotate. We tested in two scenarios, a single gripper manipulation, where one gripper stays static and the other one moves and rotates, and bimanual manipulation. The simulation time interval is 0.02s, resulting in a control frequency of 50 Hz. We have access to the positions and orientations of each segment of the DLO. The goal of the shape control task is to deform the DLO into a desired shape from a random starting configuration, as shown in Fig. 3. The goal is fixed through the training process for both the model-free RL and the online model learning methods.

Difference in shape L_{shape} is measured by Euclidean distance between segments positions $\boldsymbol{p}_{\text{goal}}$ of a DLO in target shape and achieved segments positions $\boldsymbol{p}_t$.

$$L_{\text{shape}}(\boldsymbol{p}_t, \boldsymbol{p}_{\text{goal}}) = \|\boldsymbol{p}_t - \boldsymbol{p}_{\text{goal}}\|_2 \tag{11}$$

B. Reinforcement Learning Baselines

In this paper, we test our method with PILCO [19], Twin Delayed Deep Deterministic Policy Gradient (TD3) [20], and Soft Actor-Critic (SAC) [21] in DLO shape control tasks for comparison. In previous work [43], a policy gradient method, Deep Deterministic Policy Gradient (DDPG) [33], easily solves learning 1D shape control of elastic and elastoplastic DLOs. We use a similar RL formulation and try to solve 3D shape control by PILCO, TD3, and SAC.

In the DLO shape control task, the action — $\mathcal{A} \in \mathbb{R}^6$ for single gripper manipulation, or $\mathcal{A} \in \mathbb{R}^{12}$ for bimanual manipulation — is velocity and angular velocity of the single or both grippers. Each action is rescaled to $a \in [-1, 1]$. We rescale predicted actions from the policy into viable velocity commands according to the predefined maximum accelerations and velocities. The dense reward function is defined as follows:

$$r(\boldsymbol{p}_t, \boldsymbol{a}_t) = -L_{\text{shape}}(\boldsymbol{p}_{t+1}, \boldsymbol{p}_{\text{goal}}) = -\|\boldsymbol{p}_{t+1} - \boldsymbol{p}_{\text{goal}}\|_2 \tag{12}$$

Similar to [43], we clip the reward function and $r \in [-1.5, 0]$

For our experiment, we use TD3 and SAC in *Stable-Baselines3*, a set of implementations of RL algorithms in PyTorch [44].

C. Details of Online Model Learning with MPC

In Algorithm 1, we set the maximum iteration times to 20, $max\_iter = 20$. In each iteration, we set 50 epochs for updating the data-driven model $\hat{f}_\theta$ and execute $N = 50$ trails with $T = 200$ time steps each. We select time horizon $H = 5$ for the MPC.

For model learning, we implement the data-driven model $\hat{f}_\theta$ in PyTorch [45] and choose Adam [46] as the optimizer with a fixed learning rate 1×10^{-4}. Regarding details in neural network structure, the relation-centric network, f_e, and the object-centric network, f_v, are implemented as 4-layer fully connected neural networks with hidden layers of 150-unit width. ReLU activation is used except for the final output of $\hat{f}_\theta$. For the recurrent model, we use a 2-layer bi-LSTM model with 150-dimensional hidden state vectors.

D. Results

We first compared our online model learning with MPC with a model-based RL algorithm (PILCO) and two model-free RL algorithms (SAC and TD3) concerning sample efficiency. Fig. 4 shows the cumulative rewards in the single gripper manipulation and Fig. 5 shows the rewards in bimanual manipulation. We train different RL agents for two scenarios respectively while we also directly test the control performance using the same DLO models trained in single gripper manipulation in bimanual manipulation shape control tasks. From the experiment results, it is clear that our online model learning approach outperforms SAC and TD3 with better sample efficiency. It also demonstrates that our online model learning method is versatile. Due to the data-driven model we used in online model learning, we can directly use the trained model from single gripper manipulation tasks to bimanual manipulation tasks. Both figures imply the model learned online with MPC performs better (higher cumulative reward) compared to SAC and TD3 with the same amount of samples, while PILCO cannot find a proper model and a suitable policy to finish the tasks.

Since two model-free reinforcement learning agents are trained in a scenario where a DLO shape control task has the same goal throughout the training, it is expected that the control performance will drop if we choose random goals for the agents, as shown in Fig. 6. However, our online model learning with MPC can still achieve acceptable performance despite performance drops compared to Fig. 4. This demonstrates the proposed online model learning achieves a better generalization with a small number of interaction samples.

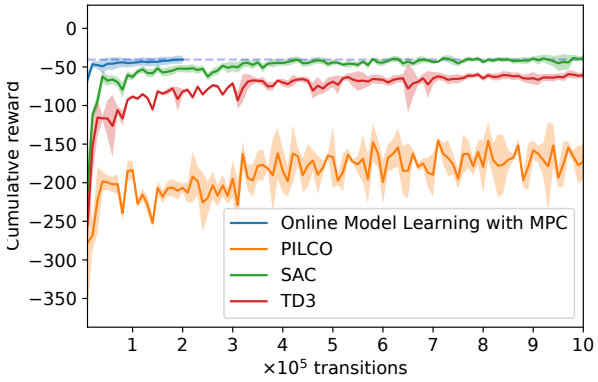

Fig. 4. The cumulative reward of trials for the DLO shape control task by a single gripper through PILCO, SAC, TD3, and, online model learning with MPC. The shaded areas represent the standard deviation.

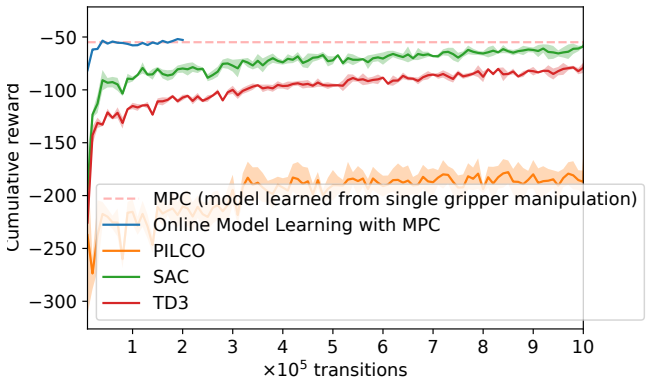

Fig. 5. Final rewards at the end of trials for DLO shape control task by two grippers through PILCO, SAC, TD3, and online model learning with MPC. The shaded areas represent the standard deviation.

Finally, we also compare online model learning with MPC with offline model learning following the method from our previous work [13]. The cumulative reward is shown in Fig. 7. With the same amount of samples for training a data-driven model, online model learning with MPC boosts the control performance at the beginning with a small number of samples. It achieves higher cumulative rewards over offline model learning within 90 thousand samples before the rewards drop slowly as more data is collected. We hypothesize that the performance degradation might result from similar on-policy data collected in each trial overfitting the data-driven model.

VI. Conclusion

We presented an online model learning framework with model predictive control (MPC). It learns a data-driven model for DLO shape control tasks using a small number of samples. Compared to prior work, we address a DLO shape control task in 3D space, while other implementations are mostly limited to a rope in 2D space. Our proposed online model learning method is based on a neural network model specifically designed for DLO. Due to the learned model, the agent can deform a DLO into the target shape from a

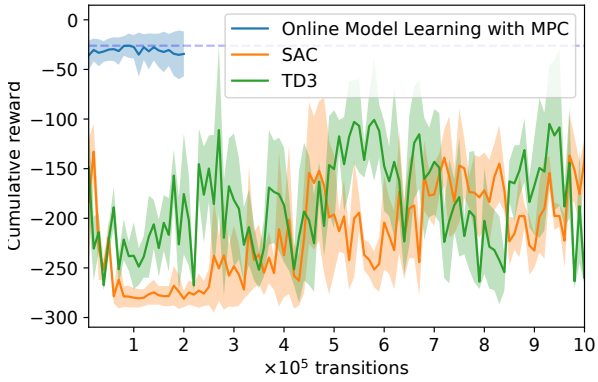

Fig. 6. The cumulative reward of trials for DLO shape control tasks by a single gripper through PILCO, SAC, TD3, and online model learning with MPC. The shaded areas represent standard deviation. Here, target shapes are random sampled which are different from the ones used in training.

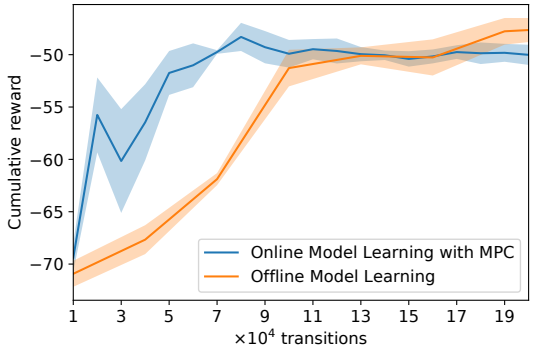

Fig. 7. The cumulative reward of trials for DLO shape control task by a single gripper through MPC using the models from online model learning and offline model learning. The shaded areas represent the standard deviation.

random starting configuration and also deform a DLO into other shapes which are not trained during online learning. The model learned in single gripper manipulation tasks can be directly used for bimanual manipulation tasks. In contrast, other model-based RL and model-free RL algorithms tested in the paper consistently failed to reach a random goal and need task-specific retraining for bimanual tasks.

We expect it would work for real robot applications given a proper DLO state tracking method and safety measures. However, we leave it as future work. Action selection solved with stochastic gradient descent shows the advantage of differentiable data-driven models, but this approach is slightly computationally expensive for real systems. In future work, we will focus on improving efficiency.

References

[1] P. Jiménez, "Survey on model-based manipulation planning of deformable objects," *Robotics and computer-integrated manufacturing*, vol. 28, no. 2, pp. 154–163, 2012.

[2] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[3] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, p. eabd8803, 2021.

[4] W. Wang, D. Berenson, and D. Balkcom, "An online method for tight-tolerance insertion tasks for string and rope," in *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 2488–2495.

[5] E. Yoshida, K. Ayusawa, I. G. Ramirez-Alpizar, K. Harada, C. Duriez, and A. Kheddar, "Simulation-based optimal motion planning for deformable object," in *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*. IEEE, 2015, pp. 1–6.

[6] D. K. Pai, "Strands: Interactive simulation of thin solids using cosserat models," in *Computer Graphics Forum*, vol. 21, no. 3. Wiley Online Library, 2002, pp. 347–352.

[7] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1141–1150, 2007.

[8] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE, 2017, pp. 2146–2153.

[9] N. Alvarez, K. Yamazaki, and T. Matsubara, "An approach to realistic physical simulation of digitally captured deformable linear objects," in *IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. IEEE, 2016, pp. 135–140.

[10] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, *et al.*, "Interaction networks for learning about objects, relations and physics," in *Advances in neural information processing systems*, 2016, pp. 4502–4510.

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

[12] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake, "Propagation networks for model-based control under partial observation," in *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE, 2019, pp. 1205–1211.

[13] Y. Yang, J. A. Stork, and T. Stoyanov, "Learning to propagate interaction effects for modeling deformable linear objects dynamics," in *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE, 2021, pp. 1950–1957.

[14] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," *arXiv preprint arXiv:1707.01495*, 2017.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[16] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and trends in Robotics*, vol. 2, no. 1-2, pp. 388–403, 2013.

[17] M. Servin and C. Lacoursiere, "Rigid body cable for virtual environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 783–796, 2008.

[18] M. Servin, C. Lacoursiere, and N. Melin, "Interactive simulation of elastic deformable materials," in *SIGRAD 2006. The Annual SIGRAD Conference; Special Theme: Computer Games*, no. 019. Citeseer, 2006.

[19] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 465–472.

[20] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[22] N. Umetani, R. Schmidt, and J. Stam, "Position-based elastic rods," in *ACM SIGGRAPH 2014 Talks*, 2014, pp. 1–1.

[23] T. Kugelstadt and E. Schömer, "Position and orientation based cosserat rods." in *Symposium on Computer Animation*, 2016, pp. 169–178.

[24] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 479–484.

[25] D. Navarro-Alarcon, H. M. Yip, Z. Wang, Y.-H. Liu, F. Zhong, T. Zhang, and P. Li, "Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 429–441, 2016.

[26] D. Navarro-Alarcon, Y.-h. Liu, J. G. Romero, and P. Li, "On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments," *International Journal of Robotics Research*, vol. 33, no. 11, pp. 1462–1480, 2014.

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

[28] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, no. 4, pp. 319–340, 2011.

[29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[30] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *ICLR*, 2019.

[31] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," *arXiv preprint arXiv:2002.09405*, 2020.

[32] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International conference on machine learning*. PMLR, 2016, pp. 2829–2838.

[33] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[34] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[35] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Robotics: Science and Systems VII*, vol. 7, 2011, pp. 57–64.

[36] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[37] H. Han, G. Paul, and T. Matsubara, "Model-based reinforcement learning approach for deformable linear object manipulation," in *IEEE Conference on Automation Science and Engineering*. IEEE, 2017, pp. 750–755.

[38] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1701–1710.

[39] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 7559–7566.

[40] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[41] R. Laezza and Y. Karayiannidis, "Reform: A robot learning sandbox for deformable linear object manipulation," *Proc. of the IEEE International Conference on Robotics and Automation*, 2021.

[42] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[43] R. Laezza, R. Gieselmann, F. T. Pokorny, and Y. Karayiannidis, "Shape control of elastoplastic deformable linear objects through reinforcement learning," *Proc. of the IEEE International Conference on Robotics and Automation*, 2021.

[44] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980