Learning to Propagate Interaction Effects for Modeling Deformable Linear Objects Dynamics

Yuxuan Yang, Johannes A. Stork, and Todor Stoyanov

***Abstract*— Modeling dynamics of deformable linear objects (DLOs), such as cables, hoses, sutures, and catheters, is an important and challenging problem for many robotic manipulation applications. In this paper, we propose the first method to model and learn full 3D dynamics of DLOs from data. Our approach is capable of capturing the complex twisting and bending dynamics of DLOs and allows local effects to propagate globally. To this end, we adapt the interaction network (IN) dynamics learning method for capturing the interaction between neighboring segments in a DLO and augment it with a recurrent model for propagating interaction effects along the length of a DLO. For learning twisting and bending dynamics in 3D, we also introduce a new suitable representation of DLO segments and their relationships. Unlike the original IN method, our model learns to propagate the effects of local interaction between neighboring segments to each segment in the chain within a single time step, without the need for iterated propagation steps. Evaluation of our model with synthetic and newly collected real-world data shows better accuracy and generalization in short-term and long-term predictions than the current state of the art. We further integrate our learned model in a model predictive control scheme and use it to successfully control the shape of a DLO. Our implementation is available at `https://gitsvn-nt.oru.se/ammlab-public/in-bilstm`.**

I. Introduction

Many robotic manipulation applications such as inserting and threading a needle [1], [2], surgical suturing [3], [4], or controlling the shape of a rope [5], [6] rely on deformable linear object (DLO) dynamics [7]. Modeling DLO dynamics in 3D space, even without contact with the environment, is challenging due to complex twisting and bending dynamics. As seen in Fig. 1, DLOs can not only move in all three directions in space, but additionally bend along their length and twist around their axis, while building up and releasing tension due to external forces. For this reason, accurate and robust modeling of DLO dynamics is still a key problem in robotics and subject to ongoing research [8].

In this paper, we present a novel, data-driven model for learning full 3D dynamics of DLOs from data. Learning DLO dynamics from data [9] instead of using analytical physics-based models [10] has an advantage that no analytical parameters have to be identified, computation is often much faster, and learned differentiable models can be used in model-based control methods. Challenges are that the model has to capture the complex twisting and bending dynamics and has to allow the effects of manipulating one segment of the DLO to propagate across all segments.

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 2017-02205, and was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

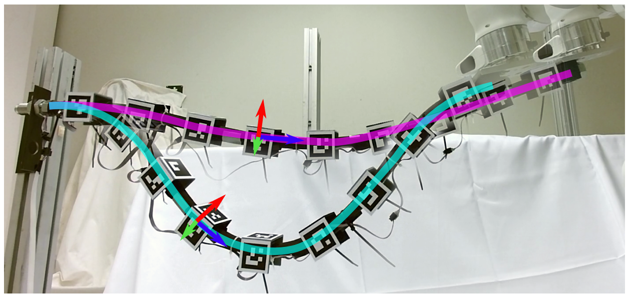
Fig. 1. DLOs have complex dynamics with bending (along the DLO) and twisting (around the internal axis) that pose challenges to model learning. Here, the indicated attached frame moves in 3D space and rotates around all three axes due to a manipulation action applied at a distant segment. Equally distributed markers on the DLO are for estimating the DLO's state.

For our model, we extend prior work and combine the interaction network (IN) [11] method, which learns local interaction dynamics between neighboring segments in a DLO, with the idea of using a recurrent model [9] for learning to propagate these local interaction effects to other segments along the length of a DLO. The IN method implements a physics-derived inductive bias which assumes that overall dynamics is made up of local interaction effects [12], while the recurrent model allows local interaction effects to reach distant segments within one time step. The latter is particularly important for modeling stiff DLOs and otherwise difficult to achieve with discrete time steps.

Our main contribution is the first data-driven model of DLO dynamics that accurately captures twisting and bending in 3D, while state-of-the-art approaches operate only on 2D [9], [11], [13] or only capture DLO position not orientation in 3D [14]. For this, we make two advances over the state of the art: 1) We introduce a new representation of DLO segments and their relationships to capture twisting and bending in 3D. 2) We extend prior work by combing two existing models of DLO dynamics to a new model, which removes the need for computationally demanding iterated interaction computation. We train and evaluate our model using both a simulation and a newly collected real-world dataset, and demonstrate experimentally that our approach can achieve better accuracy and generalization than the current state of the art. We demonstrate the utility of our approach for shape control of a DLO by integrating the learned dynamics model in a model predictive control framework. In this work, we only consider DLOs with one end fixed and the other end controlled by a robot in an otherwise empty workspace.

II. Related Work

Our work builds on existing approaches for modeling DLOs from the fields of computational physics, computer graphics, and interaction control. Current approaches are either analytical physics-based or data-driven and use different representations to model the complex dynamics of DLOs.

A. Analytical Physics-based Models

Analytical computational physics models explicitly track the forces between different segments of a DLO and numerically integrate the resulting accelerations to recover the state. These models are usually targeted at either physically accurate or visually-plausible simulations but in both cases, parameter identification and simplified dynamics can lead to errors. Physically accurate simulations are mostly based on finite element [15] or finite difference methods [16]–[19] and are particularly time-consuming because of the number of small computation steps that are required for stability and accuracy. Visually-plausible simulations are popular in computer graphics and instead aim at simple but fast computation. Position-based dynamics (PBD) [20] is such an approach and uses particles with constraints to model deformable objects and fluids. For DLOs, PBD has been extended to account for twisting [21], [22], stiffness [23], and physically meaningful parameterization [19]. We take an approach similar to PBD methods [21], [22] based on an explicitly discretized Cosserat rod [3] parameterization, however, instead of modeling constraints between segments, we learn how segments influence each other from data.

B. Data-driven Models

Data-driven models are learned directly from observations and have the potential of being accurate and computationally light enough for use in robot control and planning [24]. While theoretically well-motivated learning methods for Hamiltonian [25]–[28] and Lagrangian mechanics [29], [30] have recently been proposed, the practical relevance of these methods remains limited due to restrictive assumptions [29]. Instead, combining neural networks and compositional modeling allows learning of complex dynamics from data [31]. Our approach extends the interaction networks (IN) method [11] for modeling DLOs. INs can also model mass-spring systems, rigid and deformable bodies, and fluids [12], [13], [32]–[35]. In the context of DLOs, INs can utilize a graph network [36] to learn pair-wise local interactions between different segments of a DLO. One of the challenges of this approach is propagating the effects of interactions across the object, especially in the case of a long DLO [9]. This issue has been targeted repeatedly [13], [34], [35] by introducing different means of propagating interaction further. PropNet [13] is tested on 2D ropes and shows a better performance than original IN [11]. However, in practice, interactions between distant segments remain challenging to predict, leading to errors where local information is not sufficient for modeling dynamics [9].

Most similar to our approach are works that learn IN models of DLO dynamics in two dimensions only [13], which is too limited for most real-world robotics applications [37], or directly use a recurrent model architecture for propagating interaction effects across the DLO [9]. In contrast to [13], we consider translation-invariant dynamics in three dimensions, which increases the number of state dimensions but enables bending and twisting dynamics. Different from [9], we integrate the recurrent model into an IN model to leverage both compositional modeling of DLO dynamics from IN and learning to propagate interaction from recurrent models.

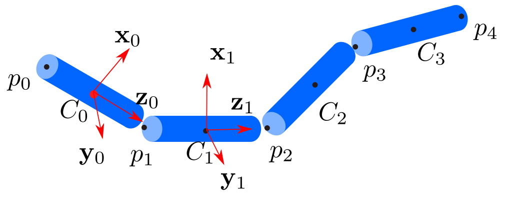

Fig. 2. Geometry of a DLO, p_i and p_{i+1} are the two endpoints of ith segment, C_i is a reference frame fixed to the center of ith segment with basis vectors $\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i$.

III. Learning to Propagate Interaction Effects

Our goal is to learn a model f for predicting the future state $\mathbf{x}_t = f(\mathbf{x}_{t-1})$ of a DLO, given the prior state $\mathbf{x}_{t-1}$, from data. In this formulation, the next state $\mathbf{x}_t$ is predicted only from the prior state x_{t-1}, which makes the choice of state representation crucial. We defer the discussion of our state and relationship representations to Sec. III-B and first consider a DLO as a finite set of neighboring segments (as seen in Fig. 2) encoded in $\mathbf{x}_t$. We refer to $\mathbf{x}_t = f(\mathbf{x}_{t-1})$ as *one-step prediction* and to the sequence $\mathbf{x}_t, f(\mathbf{x}_t), f(f(\mathbf{x}_t)), \ldots$ as a *rollout*. Accurate rollouts are particularly important for robotic manipulation with control or planning. We next outline how the IN [11] method learns DLO dynamics from data (Sec. III-A), discuss our new representation of DLO segments and their relationships (Sec. III-B) and present our new model of DLO dynamics for learning to propagate local interaction effects to reach distant segments (Sec. III-C). Finally, we explain how our model is conditioned on control inputs (Sec. III-D).

A. Interaction Networks Method

Our model of DLO dynamics is based on the IN method [11], which is a generic dynamics model for particle-based systems represented by a directed graph, $G = (V, E)$. The vertices $\mathbf{v} \in V$ represent the particles and the edges $\mathbf{e} \in E$ represent their relationships. The IN method is based on a physics-derived inductive bias. The assumption is that the overall dynamics of the particle-system is made up of local interactions between related particles according to G [12]. As seen in Fig. 3(a), the graph for a DLO would look like a chain with edges between neighboring vertices representing different segments of the DLO.

For the IN dynamics model, vertices $\mathbf{v}_i \in V$ are represented by the particle encoding $\mathbf{v}_i = (\mathbf{s}_i, \mathbf{a}_i^v)$ consisting of a particle state $\mathbf{s}_i \in \mathcal{S}$ and particle attributes $\mathbf{a}_i^v \in \mathcal{A}_v$, while directed edges $\mathbf{e}_{ij} \in E$ are represented by the relationship encoding $\mathbf{e}_{ij} = \eta\left(\mathbf{v}_i, \mathbf{v}_j, \mathbf{a}_{ij}^e\right)$ computed from the two particle encodings and some relationship attributes $\mathbf{a}_j^e \in \mathcal{A}_e$.

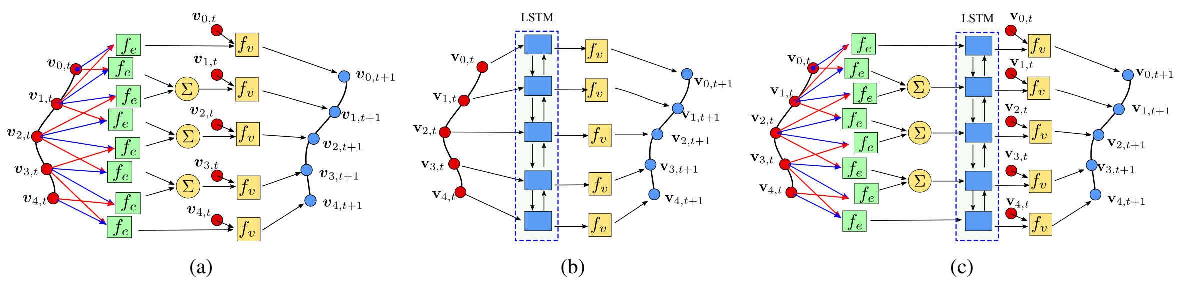
Fig. 3. Schema of (a) an interaction network (IN), (b) bidirectional LSTM, (c) IN-biLSTM (our method), for a DLO. The vertices correspond to Cosserat rod segments and neighboring segment are connected by edges. f_e is the relation-centric network and f_v is the object-centric network.

For details on representation and encodings see Sec. III-B below. In the relationship represented by the directed edge $\mathbf{e}_{ij}$, $\mathbf{v}_i$ is the receiver and $\mathbf{v}_j$ the sender. The attribute spaces $\mathcal{A}_v$ and $\mathcal{A}_e$ can be used to express material properties or different interaction types. The dynamics model f learned by the IN method, therefore, maps a state

$$\mathbf{x_{t-1}} = (\{\mathbf{v}_{i,t-1} | \mathbf{v}_i \in V\}, \{\mathbf{e}_{ij,t-1} | \mathbf{e}_{ij} \in E\}) \quad (1)$$

consisting of particle encodings $\mathbf{v}_{i,t-1}$ and relationship encodings $\mathbf{e}_{ij,t-1}$ at time step $t-1$ to the next state $\mathbf{x}_t$ with particle encodings $\mathbf{v}_{i,t}$ and relationship encodings $\mathbf{e}_{ij,t}$. While in general the attributes and also the graph G can change, e.g. to model self-collision [11], [13], [34], [35], we only describe the case where everything but the particle states $\mathbf{s}_i$ is kept constant.

The IN method learns two mappings f_e and f_v to construct the dynamics model f from above. As seen in Eq. (2), the mapping f_e maps relationship encodings to a feature space and the features belonging to the same receiver are aggregated. In Eq. (3), the mapping f_v maps aggregated features, $\mathbf{k}_i$, and particle encodings to predict future particle encodings (equal to particle states if the attributes are constant).

$$\mathbf{k}_i = \sum_{\mathbf{e}_{ij} \in E} f_e(\mathbf{e}_{ij,t-1}) \quad (2)$$

$$\hat{\mathbf{v}}_{i,t} = f_v(\mathbf{v}_{i,t-1}, \mathbf{k}_i) \quad (3)$$

The mappings f_e and f_v are referred to as *relation-centric network* and *object-centric network*, respectively, in [11].

Fig. 3(a) illustrates the compositional structure of f and shows that the overall dynamics modeled by f are made up from local interactions learned by f_e and f_v. More clearly, all dynamics are expressed as particles changing their state due to influences from related particles. While this inductive bias is intentional, it makes capturing global effects difficult and therefore methods for propagating local influences further have been proposed [13], [34], [35]. In Sec. IV we compare against such methods and demonstrate experimentally that these are insufficient for the purpose of capturing the dynamics of a DLO. Moreover, if attributes are kept constant, the model f has the previous particle states $\mathbf{s}_{i,t-1}$ which directly determine the following particle states $\mathbf{s}_{i,t}$. Obviously, the representation for $\mathbf{s}_i$ and the relationships encoding function η are therefore important for the range of dynamics the model can capture. In the next section, we present our representations for learning the complex twisting and bending dynamics in 3D.

B. Representation of DLO in 3D

Our representation is based on Cosserat theory [3], which defines DLOs as a sequence of orthonormal frames on a smooth curve where always one basis vector is parallel to the tangent. Concretely, we employ a discrete formulation of Cosserat rods for DLOs [15], [18], [22], [38] that represents DLOs as a series of N linear cylindrical segments, as depicted in Fig. 2. Each segment has an individual coordinate frame C_i at the center and neighboring segments share endpoints.

Prior approaches largely learn DLO dynamics in the 2D plane [9], [11], [13], which makes $\mathcal{S} = \mathbb{R}^2$ a natural choice for the particle state space. However, for full 3D dynamics, simply extending the particle state space to $\mathcal{S} = \mathbb{R}^3$ is insufficient because this cannot capture the twisting and bending seen in Fig. 1. For this, the orientations of the segment frames need to be reflected in particle states. We explore two different options to achieve this.

Particle State: Position and Quaternion. In the first option, each segment corresponds to one of N particles in the dynamic model and neighboring segments are connected by edges in both directions. The particle state is defined as $\mathbf{s}_i^{PQ} = (\mathbf{p}_i^C, \mathbf{q}_i^C, \dot{\mathbf{p}}_i^C, \boldsymbol{\omega}_i)$, with $\mathbf{p}_i^C, \dot{\mathbf{p}}_i^C \in \mathbb{R}^3$ being the position and velocity of the segment's coordinate frame C_i. Symbols $\mathbf{q}_i \in \mathbb{H}$ and $\boldsymbol{\omega}_i \in \mathbb{R}^3$ are the orientation of the segment's reference frame as a unit quaternion and angular velocity, respectively. We denote this encoding as PQ (Position and Quaternion).

While the PQ encoding is sound in terms of representation power, there are several issues that make it a problematic choice. First, the discontinuities in 3D/4D representations of orientation (e.g., Euler angles or quaternions) can result in a step response when calculating loss functions and back-propagating gradients in neural network function approximators [39]. In addition, the PQ encoding overparameterize the real DLO state and can represent invalid DLO states in which endpoints of neighboring segments are not in the same location.

Particle State: Position and Twist Angle. Our solution is inspired by parameterization commonly used in robotics for Kinematic chains and ensures that endpoints neighboring segments always overlap. For this, we impose a constraint on the parameterization that fixes the Z-axis of each segments' reference frame to always point along the line $\overline{\mathbf{p}_{i-1}\mathbf{p}_i}$. In the dynamics model, we use the $N+1$ segment endpoints for particles with state $\mathbf{s}_{i,t}^{PT} = (\mathbf{p}_i, \psi, \dot{\mathbf{p}}_i, \dot{\psi}_i)$, where $\mathbf{p}_i, \dot{\mathbf{p}}_i \in \mathbb{R}^3$

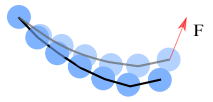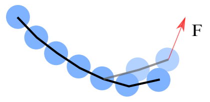

Fig. 4. A force is applied to one end of a DLO, in reality (left), a sequence of particles will move. By contrast, when IN is used to model it (right), only the nearest particle will move

are the position and velocity of individual endpoints. With one axis fixed by the endpoint positions, the twist angle $\psi_i \in \mathbb{R}$ is then sufficient to encode the full orientation of C_i. This position and twist angle (PT) encoding is a minimum explicit description for DLOs [40]. We convert between rotation matrix and twist representations with a straight-forward approach using the tilt-and-torsion Euler angle formulation [41]. Note that for numerical reasons we encode the overall accumulated twist along the chain.

Relationship Representation. It is common to define the relationship encoding function η as a concatenation of the particle encodings of both receiver and sender, i.e. $\mathbf{e}_{ij} = \left(\mathbf{v}_i, \mathbf{v}_j, \mathbf{a}^e_{ij}\right)$ [9], [11], [13]. We call this absolute encoding. However, this encoding is not translation invariant and the dynamics model f cannot learn to predict identical DLO dynamics at different positions without additional training data. Instead, we define translation invariant relationship encoding (or relative encoding) functions η. For the PQ representation, we define

$$\mathbf{e}_{ij} = (\mathbf{p}^C_i - \mathbf{p}^C_j, \bar{\mathbf{q}}^C_i \mathbf{q}^C_j, \dot{\mathbf{p}}^C_i - \dot{\mathbf{p}}^C_j, \boldsymbol{\omega}_i - \boldsymbol{\omega}_j) \tag{4}$$

where $\bar{\mathbf{q}}$ is the conjugate of $\mathbf{q}$. For the PT representation, we define

$$\mathbf{e}_{ij} = (\mathbf{p}_i - \mathbf{p}_j, \psi_i - \psi_j, \dot{\mathbf{p}}_i - \dot{\mathbf{p}}_j, \dot{\psi}_i - \dot{\psi}_j) \tag{5}$$

In our experiments in Sec. IV-C and IV-D, we find that the latter approach leads to better results.

C. *Recurrent Model of Interaction Effect Propagation*

In the original IN, only local information in the graph is considered at every time step which makes learning longer-reaching interactions difficult. For example, as shown in Fig. 4. To improve the IN effect propagation, other works [13], [35] introduce iterating procedures for instantaneous effect propagation in a single time step. The number of iterations is a hyperparameter which needs to be tested and decided case by case. More iterations make the training process harder because of heavier computation burden and the more complex model. Therefore, this approach is not suitable for long objects like DLOs.

Inspired by the work of Yan et al. [9], depicted in 3(b), we use a bidirectional LSTM (bi-LSTM) [42] for effect propagation. Here LSTM is used spatially, which perfectly fits the linear DLO structure. Each vertex in a DLO is assumed to have an identical dynamics so we can use an LSTM cell [43] recurrently along the chain. Instead of setting an iteration hyperparameter, the LSTM learns how the effects propagate along the DLO from data. In position-based dynamic models of DLO, a bilateral interleaving order is used to improve stability [21], [22]. Similarly, we choose bi-LSTM where the propagation runs in both directions simultaneously.

The structure of the model is shown in Fig. 3(c). We propagate aggregated features $\mathbf{k}_i$ of each vertex to all the other vertices using an m-layer LSTM, f_{LSTM}, along two directions.

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

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

where $(\mathbf{h}^l_0, \mathbf{c}^l_0) = f_{LSTM}(\mathbf{k}_0, \mathbf{0}, \mathbf{0})$ and for the last vertex $(\mathbf{h}^r_N, \mathbf{c}^r_N) = f_{LSTM}(\mathbf{k}_N, \mathbf{0}, \mathbf{0})$, $\mathbf{h}_i$ and $\mathbf{c}_i$ are the hidden state vectors and the cell state vector in the LSTM with superscripts l and r indicating the direction of propagation.

We concatenate the last-layer hidden state vectors $\mathbf{h}^{l,m}_i$ and $\mathbf{h}^{r,m}_i$ as propagated features which have effect information from all vertices and use them as the second input of f_v in Eq. (3) instead of the aggregated features, $\mathbf{k}_i$.

D. *Control-conditioned Dynamics*

Above, we considered the dynamics model f without any external influences on the DLO such as control. However, for manipulation we want to predict future DLO states when segments of the DLO are controlled by a robot or physically constrained in place. Both our model and the IN method allow for conditioning the model on control by exploiting different constant particle and relationship attributes. Here we detail the approach for a case in which the first particle is fixed to a wall and the last particle's state is controlled by a robot as used in our experiments in Sec. IV.

First, we divide the particles V into V_C (externally controlled by the robot or the environment), and all remaining free particles $\bar{V}$. The control conditioned dynamics model f_C inputs the state $\bar{\mathbf{x}}_{t-1}$ of the free particles $\bar{V}$ and the controlled particles' (externally forced) states as control input. The model f_C then constructs a new and complete state $\mathbf{x}^*_{t-1}$ as the union of the free particle states and the forced controlled particle states, where an indicator variable in the particle attributes $\mathbf{a}^v_i$ is used to determine if a particle is free to move. Finally, it applies the general dynamics model f to the complete state $\mathbf{x}^*_{t-1}$ for prediction. To learn control-conditioned dynamics from data without observed control commands, the controlled particles' input and target states can be set to the observed values in the next time step, which is what we do in Sec. IV.

IV. Experiments and Results

A. *Data Generation and Collection*

Synthetic Data. We generate data from simulation using the physics-engine Bullet [44]. In the simulation, a 3.36-meter DLO is modeled by 16 segments (cylinders), where each pair of segments is connected by a special 6D spring, which enables stretch, shear, bend, and twist deformations. We generate 3000 rollouts over 300 time steps each, by keeping one end of the DLO fixed and applying random controls to the other end. The interval between steps is $1/60$s.

Real-world Data. We also collect motion data for a 0.9-meter real DLO (a stiff hydraulic fluid hose) to validate the effectiveness of the model for a real DLO. Similar to the synthetic data, one end of the DLO is fixed and the other end is manipulated by a Franka Panda robot arm, as shown in Fig. 1. The DLO is discretized into nine segments on the centers of which we mounted holders with attached Aruco markers [45]. The poses of the markers are estimated from images captured by a Microsoft Kinect. The segments' poses are transformed from corresponding markers' poses. The images are captured at 20 frames per second. 1000 rollouts data are collected and each rollout lasts 4 seconds. We use a Kalman Filter [46] to filter out noise in the observed markers' positions and orientations over time.

B. Training Details

We implement the model in PyTorch. Adam [47] is the optimizer with a fixed learning rate 1×10^{-4}. For training on synthetic data, the batch size is 512, while for real-world data, the batch size is 128. The data are split into training and validation with a 90-10 split. All models are trained on a *one-step prediction* task with an L^2 loss on the respective particle state encoding for free particles in subsequent time steps. In order to make the models more robust to noise in the inputs (as can be expected when model predictions are fed-back as inputs during *rollout*), we augment the training data with additive white Gaussian noise.

The mapping f_e is implemented as a 4-layer fully connected neural network, while f_v is decomposed into a state encoder and an effect predictor, same as in Interaction Networks. Each of these networks is 4-layer and fully connected. In all cases we use hidden layers of width 150 units. ReLU activation is used except for the output of the effect predictor. 2-layer bi-LSTM model with 150-dimensional hidden state vectors is used unless otherwise specified.

C. Long-term Prediction

When a learned model is used in simulation or model predictive control, long-term prediction performance is important. To test this performance, we use learned models to predict a sequence of a DLO's future state through 300 iterations which is equivalent to a 5-second time span.

For the long-term prediction results, errors consist of error in position and error in orientation. Position error is the average Euclidean distance between particles' predicted positions and corresponding ground truth during a rollout. The difference in orientation is computed by [48]

$$\Phi(R_1, R_2) = \| \log(R_1 R_2^T) \| \tag{8}$$

where matrix logarithm $\log(\mathbf{R})$ gives the skew-symmetric matrix which encodes rotation axis and angle, and $||.||$ gives the absolute value of the rotation angle from R_2 to R_1. The value is in the range $[0, \pi)$. The errors in orientation are the average difference between predicted orientations and ground truth orientations. Data with PT representation are converted to PQ representation for comparison.

As shown in Table I, models of PropNet, in general, generate larger errors. The models with 15-step propagation, in theory, should propagate effect through the whole DLO leading to a better performance, but the improvement in long-term prediction is limited while more time is needed in training and prediction because of more layers in the models.

IN-biLSTM (rel-PT) and bi-LSTM (abs-PQ) have similar good performance with respect to the RMSE and median error in position and orientation. However, the latter can predict unrealistic states with some parts of a DLO disconnecting as shown in Fig. 5(b). By contrast, although the former generates long-term prediction results with large errors occasionally, the results keep being visually plausible, as shown in Fig. 5(c). Due to the relative coding, our approach IN-biLSTM(rel-PT) is translation invariant, while the performance of a baseline bi-LSTM(abs-PQ) drops when a constant translational offset is added to every segment of a DLO, as shown in Table II.

Our approach, IN-biLSTM(rel-PT), is tested in learning the dynamics of a real DLO. For 80-step (equals to 4s) forward prediction test on 100 cases, the results are RMSE of $2.0943 \times 10^{-2}\mathrm{m}$ and median error of $1.6976 \times 10^{-2}\mathrm{m}$ in position and RMSE of $2.2615 \times 10^{-1}\mathrm{rad}$ and median error of $2.1795 \times 10^{-1}\mathrm{rad}$ in orientation. An example is shown in Fig. 6.

TABLE I
LONG-TERM PREDICTION PERFORMANCE OF DIFFERENT MODELS

Method [a]	m [b]	Position error (m)		Orientation error (rad)	
		RMSE	Median	RMSE	Median
PropNet(rel-PQ)	3	0.4952	0.4145	0.7008	0.5832
PropNet(rel-PT)	3	0.2880	0.1994	0.4542	0.3281
PropNet(rel-PQ)	15	0.5278	0.2286	0.4774	0.2743
PropNet(rel-PT)	15	0.2296	0.1031	0.3772	0.1902
bi-LSTM(abs-PQ)	2	**0.1266**	**0.0475**	**0.1844**	**0.0637**
bi-LSTM(abs-PT)	2	0.2122	0.1282	0.3283	0.2046
IN-biLSTM(rel-PQ)	2	0.4289	0.2313	0.3685	0.1871
IN-biLSTM(rel-PT)	2	**0.1346**	**0.0483**	**0.2268**	**0.0762**

[a] PT and PQ indicate the representation the model trained on. The relative encoding and absolute encoding are abbreviated as rel and abs, respectively.
[b] m represents propagation steps in PropNet or the number of LSTM layers in LSTM-related methods.

TABLE II
PERFORMANCE OF BI-LSTM AFFECTED BY TRANSLATIONAL OFFSETS

σ [a]	Position error (m)		Orientation error (rad)	
	RMSE	Median	RMSE	Median
0	0.1266	0.0475	0.1844	0.0637
0.1	0.1456	0.0903	0.1918	0.0786
0.3	0.2728	0.2531	0.2572	0.1754
1.0	0.7715	0.7581	0.5539	0.4750

[a] The standard deviation in $\mathcal{N}(0, \sigma^2)$ where random translational offsets (in meter) are sampled from.

Using a trained network for forward prediction is faster than the analytic physics simulator used (Bullet). On a laptop with Inter i9-9980HK and GTX1650, a 5-second DLO simulation is generated in about 2.5 seconds using the C++ version of Bullet. By contrast, 1.2 seconds are needed for the same simulation based on a learned model of IN-biLSTM(rel-PT).

D. Ablation Studies

Ablation studies are provided to show how the number of LSTM layers and choice of DLO representation influence the final performance. More layers of LSTM have more parameters and, in general, have larger capacity to approximate

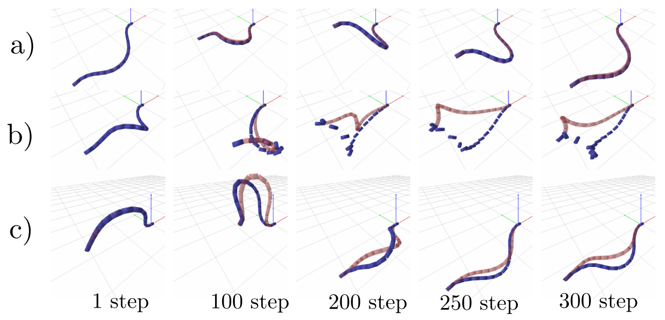

Fig. 5. Qualitative results in simulation. The objects in blue are forward predictions, the objects in semi-transparent red are the ground truth. Time interval of 1/60s is between steps. a) a rollout prediction by IN-bi-LSTM(rel-PT); b) a rollout prediction of largest error in validation dataset by bi-LSTM(abs-PQ); c) a rollout prediction of largest error in validation dataset by IN-biLSTM(rel-PT).

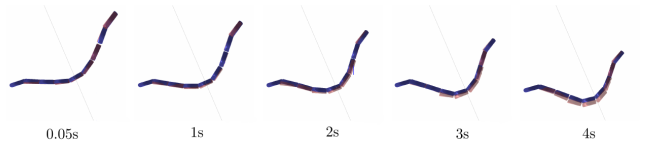

Fig. 6. Qualitative results in forward simulation of a real DLO. The ground truth is in semi-transparent red while forward prediction is in blue.

complex dynamics. We test IN-biLSTM with 1, 2, 3 layers of bi-LSTM cells in long-term predictions, the results of which are shown in Table III. As expected, more layers of LSTM cells can approximate the dynamics of a DLO better. Although the three-layer model has the best results with the smallest errors, in practice the improvement is not notable and comes at a cost of increased computational demand both in training and prediction. As a trade-off between efficiency and accuracy, we choose a two-layer bi-LSTM as a backbone for the shape control task in Sec. (IV-E).

To validate how different DLOs representations and relationship encodings influence the model performance in long-term prediction, we tested four different options, as shown in Table IV. Results indicate our proposed relative PT encoding works the best in long-term forward prediction. Compared to PT representation, PQ representation which is redundant requires models to learn more constraints. Relative encoding helps the model approximate the dynamics better in the case of PT representation. The inductive bias inherent in this encoding: namely, that the dynamics are invariant to spatial position; likely results in predictions that are more consistent with the underlying physics.

E. Shape Control Task Using a Learned Model

Finally, we evaluate our models in a shape control task: i.e., achieving a desired DLO state. The learned neural network models are differentiable, thus we can use them directly in model predictive control. Control inputs are optimized by minimizing a loss between a target state and the state from forward simulation. In the task, the DLO has one end fixed and has the other end controlled. Similar to Li et al. [13], we choose model predictive control (MPC) with shooting methods for the control task. Given an initial state $\mathbf{x}_0$, a dynamics model f and a goal state $\mathbf{x}_g$. Through forward simulation $\mathbf{x}_{t+1} = f(\mathbf{x}_t)$, we can have a set of predicted

TABLE III
LONG-TERM PREDICTION PERFORMANCE OF IN-BILSTM(REL-PT) WITH DIFFERENT NUMBERS OF LSTM LAYERS

The number of layers	Position error (m) RMSE	Median	Orientation error (rad) RMSE	Median
1	0.4440	0.0735	0.3894	0.2277
2	0.1346	0.0483	0.2268	0.0762
3	0.1137	0.0410	0.1862	0.0656

TABLE IV
LONG-TERM PREDICTION PERFORMANCE OF IN-BILSTM ON DATA WITH DIFFERENT ENCODINGS

Method	Position error (m) RMSE	Median	Orientation error (rad) RMSE	Median
IN-biLSTM(rel-PT)	**0.1346**	**0.0483**	**0.2268**	**0.0762**
IN-biLSTM(abs-PT)	0.3071	0.1162	0.3684	0.1794
IN-biLSTM(rel-PQ)	0.4289	0.2313	0.3685	0.1871
IN-biLSTM(abs-PQ)	0.2040	0.1231	0.2390	0.1247

TABLE V
PERFORMANCE OF SHAPE CONTROL TASK

Method	Position error (m) RMSE	Median	Orientation error (rad) RMSE	Median
IN-biLSTM(rel-PT)	0.2154	0.1104	0.0253	0.0121

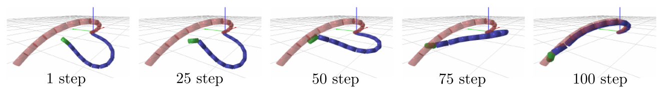

Fig. 7. Qualitative results of MPC control. The semi-transparent red DLO is the target configuration and the blue DLO is controlled by the green segment (one end of the DLO on the left side) to reach the target.

states as $\mathbf{X} = \{\mathbf{x}_i\}_{i=1\ldots t}$. Gradients with respect to the control commands can be backpropagated from loss function $L(\mathbf{x}_t, \mathbf{x}_g)$ and used for optimizing control commands by stochastic gradient descent (SGD).

Our best model, IN-biLSTM(rel-PT), is used in MPC for evaluation. We test the shape control task in simulation. To guarantee a chosen initial state and target state are valid and reachable, we randomly select them from the validation dataset. We test 100 cases and results are shown in Table V. Errors are reported by the differences between the target state and the actual state of particles. Fig. 7 shows an example.

V. CONCLUSION

We have presented the first learnable model to approximate the dynamics of DLOs in three dimensions with bending and twisting behaviors. A bi-LSTM module in the model helps learn how effects propagate along a DLO and improves the long-term prediction performance. We also demonstrated that the model approximates the dynamics of DLOs better by leveraging a special-purpose position and twist representation with relative encoding. We validated the learned dynamics models in an MPC schema for achieving a desired DLO shape. As future work, more efficient and general effect-passing methods will be explored because bi-LSTM will be time-consuming for a long chain. Another aspect is to explore more powerful inductive biases for learning from translational and rotational invariance which exists in the dynamics.

References

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

[2] 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.

[3] 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.

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

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

[6] T. Bretl and Z. McCarthy, "Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 48–68, 2014.

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

[8] 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," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[9] 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.

[10] S. Javdani, S. Tandon, J. Tang, J. F. O'Brien, and P. Abbeel, "Modeling and perception of deformable one-dimensional objects," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1607–1614.

[11] 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.

[12] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[13] 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 *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1205–1211.

[14] Y. Li, T. Lin, K. Yi, D. Bear, D. L. Yamins, J. Wu, J. B. Tenenbaum, and A. Torralba, "Visual grounding of learned physical models," in *International Conference on Machine Learning*, 2020.

[15] J. Spillmann and M. Teschner, "Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2007, pp. 63–72.

[16] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," in *ACM SIGGRAPH 2008 papers*, 2008, pp. 1–12.

[17] M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, and E. Grinspun, "Discrete viscous threads," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, pp. 1–10, 2010.

[18] H. Lang, J. Linn, and M. Arnold, "Multi-body dynamics simulation of geometrically exact cosserat rods," *Multibody System Dynamics*, vol. 25, no. 3, pp. 285–312, 2011.

[19] C. Deul, T. Kugelstadt, M. Weiler, and J. Bender, "Direct position-based solver for stiff rods," in *Computer Graphics Forum*, vol. 37, no. 6. Wiley Online Library, 2018, pp. 313–324.

[20] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

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

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

[23] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: position-based simulation of compliant constrained dynamics," in *Proceedings of the 9th International Conference on Motion in Games*, 2016, pp. 49–54.

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

[25] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 379–15 389.

[26] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins, "Hamiltonian generative networks," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJenn6VFvB

[27] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou, "Symplectic recurrent neural networks," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BkgYPREtPr

[28] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia, "Hamiltonian graph networks with ode integrators," *arXiv preprint arXiv:1909.12790*, 2019.

[29] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BklHpjCqKm

[30] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020. [Online]. Available: https://openreview.net/forum?id=iE8tFa4Nq

[31] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A compositional object-based approach to learning physical dynamics," *arXiv preprint arXiv:1612.00341*, 2016.

[32] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. F. Fei-Fei, J. Tenenbaum, and D. L. Yamins, "Flexible neural representation for physics prediction," in *Advances in neural information processing systems*, 2018, pp. 8799–8810.

[33] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, "Visual interaction networks: Learning a physics simulator from video," in *Advances in neural information processing systems*, 2017, pp. 4539–4547.

[34] 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.

[35] 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.

[36] 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.

[37] A. Shah, L. Blumberg, and J. Shah, "Planning for manipulation of interlinked deformable linear objects with applications to aircraft assembly," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1823–1838, 2018.

[38] M. Grégoire and E. Schömer, "Interactive simulation of one-dimensional flexible parts," *Computer-Aided Design*, vol. 39, no. 8, pp. 694–707, 2007.

[39] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753.

[40] B. Lefevre, F. Tayeb, L. Du Peloux, and J.-F. Caron, "A 4-degree-of-freedom kirchhoff beam model for the modeling of bending–torsion couplings in active-bending structures," *International Journal of Space Structures*, vol. 32, no. 2, pp. 69–83, 2017.

[41] I. Bonev, D. Zlatanov, and C. Gosselin, "Advantages of the modified euler angles in the design and control of pkms," in *2002 Parallel Kinematic Machines International Conference*. Citeseer, 2002, pp. 171–188.

[42] 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.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[44] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[45] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[46] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[47] 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

[48] D. Q. Huynh, “Metrics for 3d rotations: Comparison and analysis,” *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.