DEFORM: Differentiable Elastic Rods For Deformable Linear Objects Realistic Modeling

Yizhou Chen, Yiting Zhang, Zachery Brei, Tiancheng Zhang, Yuzhen Chen, Ram Vasudevan

***Abstract*—This paper addresses the task of modeling deformable linear objects (DLOs), such as ropes and cables, during dynamic motion over long time horizons. This task presents significant challenges due to the complex dynamics of DLOs. To address these challenges, this paper proposes Differentiable Discrete Elastic Rods For Deformable Linear Objects Realistic Modeling (DEFORM), a novel framework that combines a differentiable physics-based model with a learning framework to model DLO accurately in real-time. The performance of DEFORM is evaluated in an experimental setup involving two industrial robots and a variety of sensors. A comprehensive series of experiments demonstrate the efficacy of DEFORM in terms of accuracy, computational speed, and generalizability, compared to state-of-the-art alternatives. To further illustrate the utility of DEFORM, this paper illustrates how to incorporate the DEFORM framework within a perception pipeline, wherein observations of the DLO are occluded. Through incorporating DEFORM into this vision pipeline, this paper enables accurate and robust tracking of DLOs. Finally, this paper demonstrates the practical application of DEFORM in a 3-D shape matching task in both simulation and real-world settings and illustrates it superior performance when compared to other state-of-the-art modeling methods.**

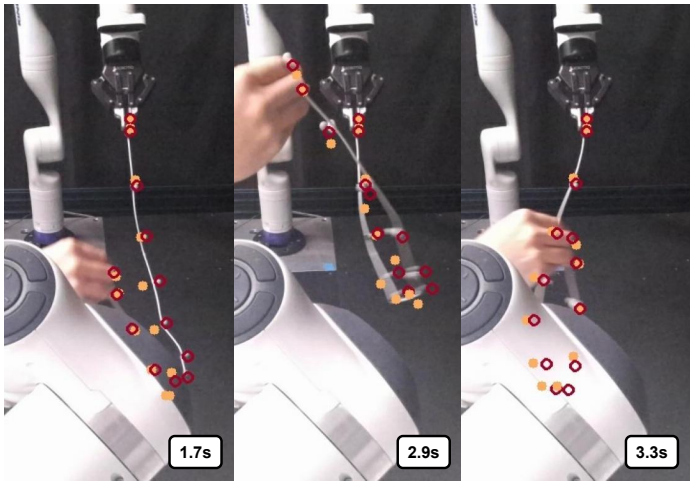


Fig. 1: Modeling DLO under dynamic manipulation: this paper introduces DEFORM, a novel framework that combines a differentiable physics-based learning model to model and predict DLO behavior accurately in real-time. The figure shows the predicted state for a DLO using an RGB-D camera under occlusion.

I. Introduction

Robotic manipulation tasks such as inserting and threading a needle, surgical suturing, or vehicle wire harness assembly [1] are challenging problems for robotic systems because they require manipulating objects such as ropes or cables [2]. These objects are often categorized as Deformable Linear Objects (DLOs). Notably, DLOs have flexible, complex, and nonlinear dynamics due to deformations. Practical applications, such as vehicle wire harness assembly, require accurate and dynamic manipulation of DLOs that require robotic manipulators to operate autonomously over long time horizons ($> 1s$). These non-trivial dynamics make accurately manipulating DLOs challenging because complex and computationally expensive models are often required for accurate prediction of behavior. For robotic manipulators to successfully complete manipulation of DLOs in practical applications, a novel modeling method capable of quick and accurate prediction of dynamic behavior of DLOs is needed.

State-of-the-art learning-based approaches for modeling DLOs have used deep neural networks (DNNs) to model interaction propagation in DLOs [3], [4]. Despite their success, these methods require large datasets of real-world manipulation data, which can be impractical to collect in many applications and are not robust to variations in the properties of a DLO. In contrast, traditional physics-based approaches typically construct a dynamic model of a DLO that is numerically integrated at run-time to predict the deformation behavior of the DLO. Due to the complexity of these models, numerical integration is expensive [5]. To achieve real-time operation, current methods increase the integration time step of these models[6]. However, this reduces the accuracy of the predictions and gives rise to numerical stability issues.

Discrete Elastic Rods (DER) is a recent physics-based approach that simplifies the modeling complexity through the use of reduced coordinate formulations [7] while providing a qualitative description of DLOs. The quantitative analysis of DER models for DLOs in real-world applications remains under explored. Further, there are three main drawbacks to DER models: 1) it is difficult to tune the parameters of these models (e.g., mass, bending and twisting modulus) to align them with real-world observations, 2) numerical errors arise from discrete numerical integration of DER models results in error for long-horizon prediction, and 3) enforcement of inextensibility in the original DER formulation does not guarantee momentum conservation which can hinder simulation stability and restrict their applicability while modeling *dynamic* DLO behavior.

This paper introduces **D**ifferentiable Discrete **E**lastic Rods **F**or Deformable Linear **O**bjects **R**ealistic **M**odeling (**DEFORM**), a novel framework that uses a differentiable physics-based learning model, with real-time inference speed, to accurately predict dynamic DLO behavior over long time horizons. The DEFORM framework, and the contributions this

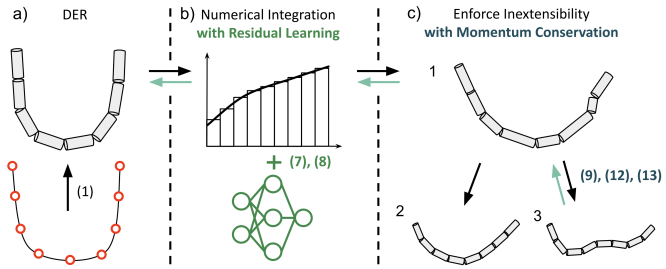


Fig. 2: Contributions of DEFORM: highlighted from light to dark green. a) DER discretizes DLOs into vertices, segments them into elastic rods, and models their dynamic propagation. DEFORM reformulates DER into Differentiable DER (DDER) that describes how to compute gradients from the prediction loss which enables automatic parameter tuning. b) To compensate for the error that arises from DER's numerical integration, DEFORM introduces residual learning via DNNs. c) $1 \rightarrow 2$: DER enforces inextensibility, but this does not satisfy classical conservation principles. $1 \rightarrow 3$: DEFORM enforces inextensibility with momentum conservation, which allows dynamic modeling while maintaining simulation stabilty.

paper makes to the DER model, are illustrated in Figure 2. To illustrate the utility of this framework, this paper shows how DEFORM can be applied to perform robust tracking of occluded DLOs in collaborative environments involving humans and robots, and DLO shape matching manipulation tasks in both simulation and the real-world. The main contributions of this paper are five-fold:

1) A reformulation of DER that makes the model differentiable with respect to the model parameters that enables model parameter identification. We refer to this extension as Differentiable Discrete Elastic Rods (DDER).
2) A novel numerical integration error compensation method that uses deep neural networks (DNNs) to improve accuracy when numerically integrating DER models.
3) A novel inextensibility enforcement algorithm for DER models that guarantees momentum conservation, enabling accurate prediction of dynamic DLO behavior.
4) An algorithm that integrates DEFORM into a perception pipeline that enables accurate tracking of DLOs during dynamic motion even in the presence of complex occlusion as illustrated in Figure 1.
5) A comprehensive set of experiments that illustrate DEFORM has superior performance in terms of accuracy, computational speed, sampling efficiency and its practicability for motion planning when compared to existing state-of-the-art approaches.

II. Related Work

This section briefly summarizes related work in DLO modeling and perception.

Physics-Based Modeling: Traditionally, DLOs have been modeled using physics-based approaches. Examples include mass-spring system models [6], Position-Based Dynamics (PBD) models [8], Discrete Elastic Rod (DER) models [7], or Finite Element Method (FEM) models [5]. Although easy to implement, the mass-spring system imposes unnecessary stiffness to enforce inextensibility that can lead to simulation instability. PBD models can be used to perform rapid inference and they are more numerically stable than the mass-spring system approach. However, PBD models are sensitive to parameter selection and have low accuracy during dynamic manipulation, as shown by our experiments. DER models have been shown to qualitatively describe the 3D behavior of DLOs and have been applied in planning and control frameworks for dynamic manipulation tasks [9]. However, their accuracy in predicting 3D behavior, particularly dynamic 3D behavior, of DLOs has not been studied. FEM models offer an accurate modeling of DLO behavior but are computational costly.

Learning-Based Modeling: The emergence of deep learning has introduced several alternatives to physics-based modeling. Latent dynamics modeling approaches, such as graph neural networks (GNN) [10] and Bidirectional LSTM (Bi-LSTM) [4], reason over vertices spatially to formulate latent dynamics of DLOs for predicting their behavior in 2-D. As we illustrate in this paper through real-world experiments, these methods are not accurate during dynamic motion. The integration of physics-based models within learning frameworks [11] is a promising avenue, but to the best of our knowledge has yet to be applied successfully to predict behavior of DLOs during dynamic motions.

Differentiable Modeling Framework: Differentiable models are computational frameworks where every operation is differentiable with respect to the input parameters. This is crucial for enabling efficient parameter adjustment via gradient-based optimization during training. DiffCloth [12] is a differentiable framework for modeling deformable objects based on the mass-spring system model. However, as mentioned earlier, mass-spring system models of DLOs tend to be numerically unstable when applied to prediction tasks, particularly over long time horizons. XPBD [8] is another differentiable modeling framework, which is built upon a PBD model and incorporates the autograd feature of PyTorch. To address these limitations, this paper proposes a differentiable DER model, called DDER, which is incorporated into a learning framework which enables better predictive performance when compared to state-of-the-art methods.

Tracking Under Occlusion: If a DLO is fully observable, then current state-of-the-art methods estimate the location of the DLO's vertices by applying a Gaussian Mixture Model (GMM), performing clustering, and then using Expectation-Maximization [13]. The output of this GMM algorithm is the mean locations in $\mathbb{R}^3$ of each of the Gaussians, which is then set equal to the vertex locations of the DLO. However, in practical applications, occlusion of DLOs during manipulation often leads to perception challenges, which complicates accurate prediction. In particular, due to occlusions, one cannot simply set the number of mixtures in the GMM equal to the number of vertices of the DLO. Doing so results in the vertices being incorrectly distributed only to the unoccluded parts of the DLO. Some of the aforementioned methods[14], [15] have been applied to perform tracking of DLOs under occlusion. Typically, this is done by leveraging short time

horizon prediction with geometric regularization using prior observations[15]. Though powerful, to work accurately, these methods require frequent measurement updates and can struggle in the presence of occlusions for long time horizons. This paper illustrates that our proposed model allows us to adapt DEFORM's long time horizon prediction capability to relax the frequency of sensor updates, which reduces the overall computational cost of tracking DLOs in the presence of occlusions.

III. Preliminaries

This section introduces the key components of DER models while describing their limitations. Note that more details about DER can be found at [7]. We also introduce existing state estimation algorithms of DLOs and discuss potential improvements.

A. Deformable Linear Object (DLO) Model

To model a DLO, we separate the DLO into an indexed set of n vertices at each time instance. The ground truth locations of vertex i at time t is denoted by $\mathbf{x}_{\mathrm{t}}^{\mathrm{i}} \in \mathbb{R}^3$ and the set of all n vertices is denoted by $\mathbf{X}_{\mathrm{t}}$. Let $\boldsymbol{M}^{\mathrm{i}} \in \mathbb{R}^{3\times3}$ denote the mass matrix of vertex i. A segment, or edge, in the DLO is the line between successive vertices, $\mathbf{e}_{\mathrm{t}}^{\mathrm{i}} = \mathbf{x}_{\mathrm{t}}^{\mathrm{i+1}} - \mathbf{x}_{\mathrm{t}}^{\mathrm{i}}$. Let $\mathbf{E}_{\mathrm{t}}$ correspond to the set of all edges at time t. Our objective is to model the dynamic behavior of a DLO that is being controlled by a robot and as a result we make the following assumption:

Assumption 1. *The first and last edge of the DLO are each held by a distinct end effector of a robot*

Note this assumption is commonly made during DLO manipulation [16]–[18]. Using this assumption, let the orientation and position of the two end effectors at time t be denoted by $\mathbf{u}_t \in \mathbb{R}^{12}$, which we refer to as the input. Let $\mathbf{U}_{1:\mathrm{T}-1}$ denote the set of inputs applied between times $t = 1$ and $t = \mathrm{T} - 1$, and let $\mathbf{X}_{1:\mathrm{T}} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{\mathrm{T}}\} \in \mathbb{R}^{\mathrm{T}\times\mathrm{n}\times3}$ denote the associated trajectory of the DLO from times $t = 1$ to $t = \mathrm{T}$.

Let Δ_t denote the difference in time between successive elements of $\mathbf{X}_{1:\mathrm{T}}$. Then, the velocity of the vertices of the DLO can be denoted as $\mathbf{V}_{\mathrm{t}} = (\mathbf{X}_{\mathrm{t}} - \mathbf{X}_{\mathrm{t\text{-}1}})/\Delta\mathrm{t}$. Finally, note that we distinguish between ground truth elements and predicted elements by using the circumflex symbol (*e.g.*, $\mathbf{X}_t$ is the ground truth set of vertices at time t and $\hat{\mathbf{X}}_{\mathrm{t}}$ is the predicted set of vertices at time t).

B. Modeling DLO with DER for Dual Manipulation

DLOs primarily exhibit three distinct modes: bending, twisting, and stretching [7]. In this paper, we focus on DLOs that are inextensible, and therefore do not experience stretching. Accurately modeling the bending and twisting modes of a DLO is challenging without introducing a large number of parameters. Unfortunately, the behavior of a model with large numbers of parameters is challenging to simulate in real-time [19], [20]. To enforce inextensibility, it is typical to apply a large stretch stiffness, but this can introduce instabilities during simulation, making these models difficult to use for prediction. To address these challenges, (1) DER introduces a single scalar $\theta_t^{\mathrm{i}} \in \mathbb{R}$ that accurately represents bending and twisting in $\mathbb{R}^3$ and (2) DER enforces inextensibility using a fast, stable projection method [21, Section 4].

1) Potential Energy to Equations of Motion: A DLO can be modeled using DER by first introducing a family of frames that are collocated with each of the vertices. In particular, each frame's relationship with the prior frame is parameterized by $\theta_t^{\mathrm{i}} \in \mathbb{R}$. An explanation for why a single scalar is sufficient to describe this rotation can be found in [7, Section 4.2]. If one is given $\mathbf{X}_t$, then one can compute θ_t^{i}. As a result, we write $\boldsymbol{\theta}_t(\mathbf{X}_t)$ to denote the vector of all θ_t^{i} at a time t. Note, we explain how to compute $\boldsymbol{\theta}_t(\mathbf{X}_t)$ before the end of this section.

Using this representation, DER is able to accurately approximate the potential energy of the DLO that arises due to bending or twisting. This potential energy is a function of the material properties of each vertex of the DLO (i.e., their mass, bending modulus, and twisting modulus) and the configuration of the DLO. If we let $\boldsymbol{\alpha}$ denote the vector of the material properties of each vertex of the DLO, then one can write down the potential energy of a particular configuration of the DLO as $P(\mathbf{X}_t, \boldsymbol{\theta}_t(\mathbf{X}_t), \boldsymbol{\alpha})$ [7, Section 4.2].

Once the potential energy expression is derived, then the restorative force experienced during deformation is equal to the negative of the gradient of the potential energy expression with respect to the vertices. Therefore, the equation of motion of the DLO is:

$$\boldsymbol{M}\ddot{\mathbf{X}}_t = -\frac{\partial P(\mathbf{X}_t, \boldsymbol{\theta}_t(\mathbf{X}_t), \boldsymbol{\alpha})}{\partial \mathbf{X}_t} \tag{1}$$

One can numerically integrate this formula to predict the velocity and position by applying the Semi-Implicit Euler method:

$$\hat{\mathbf{V}}_{t+1} = \hat{\mathbf{V}}_{\mathrm{t}} - \Delta_{\mathrm{t}} M^{-1} \frac{\partial P(\hat{\mathbf{X}}_t, \boldsymbol{\theta}_t(\hat{\mathbf{X}}_t), \boldsymbol{\alpha})}{\partial \mathbf{X}_t}, \tag{2}$$

$$\hat{\mathbf{X}}_{\mathrm{t}+1} = \hat{\mathbf{X}}_{\mathrm{t}} + \Delta_{\mathrm{t}} \hat{\mathbf{V}}_{t+1}. \tag{3}$$

One can use (2)-(3) with state-of-the-art ODE solvers to simulate the behavior of the DER model of a DLO. However, computing the gradient of the potential energy requires an expression for $\boldsymbol{\theta}_t(\mathbf{X}_t)$. To construct such an expression, DER assumes that the DLO reaches an equilibrium state between each simulation time step:

Assumption 2. *Between each step of the simulation, the DLO reaches an equilibrium state. As a result, at each time step t, one can compute*

$$\boldsymbol{\theta}_t^*(\boldsymbol{X}_t) = \underset{\boldsymbol{\theta}_t}{\arg\min}\, P(\boldsymbol{X}_t, \boldsymbol{\theta}_t, \boldsymbol{\alpha}) \tag{4}$$

To summarize, at each integration time step, one must solve (4) to evaluate the dynamics. There are several factors which make simulating this model quickly and accurately challenging. First, it may be difficult to accurately estimate the material properties of an arbitrary DLO before simulating the model. In Section.IV-A, we illustrate how to make the DER model differentiable so that one can estimate the material properties

Algorithm 1 One Step Prediction

Inputs: $\hat{\mathbf{X}}_t, \hat{\mathbf{V}}_t, \mathbf{u}_t$

$\boldsymbol{\theta}_t^*(\mathbf{X}_t) = \arg\min_{\boldsymbol{\theta}_t} P(\mathbf{X}_t, \boldsymbol{\theta}_t, \boldsymbol{\alpha})$ ▷ Section.IV-A

$\hat{\mathbf{V}}_{t+1} = \hat{\mathbf{V}}_t - \Delta_t M^{-1} \frac{\partial P(\mathbf{X}_t, \boldsymbol{\theta}_t^*(\mathbf{X}_t), \boldsymbol{\alpha})}{\partial \hat{\mathbf{X}}_t}$ ▷ (3)

$\hat{\mathbf{X}}_{t+1} = \hat{\mathbf{X}}_t + \Delta_t \hat{\mathbf{V}}_{t+1} + \text{DNN}$ ▷ Section.IV-B

Inextensibility Enforcement on $\hat{\mathbf{X}}_{t+1}$ ▷ Section.IV-C

$\hat{\mathbf{V}}_{t+1} = (\hat{\mathbf{X}}_{t+1} - \hat{\mathbf{X}}_t)\Delta_t$ ▷ Velocity Update

return $\hat{\mathbf{X}}_{t+1}, \hat{\mathbf{V}}_{t+1}$

from real-world data while simulating the model. Second, there is an undesirable tradeoff between choosing a larger time step to achieve real-time prediction and the accuracy of that prediction. In Section.IV-B, we apply residual learning to perform predictive compensation during integration which enables rapid simulation of the model without sacrificing accuracy.

2) Inextensibility Enforcement: During simulation, the DLO's length may change due to numerical errors, but this is undesirable when modeling DLOs like cables and ropes, which are inextensible. To ensure that the length of the DLO does not change during simulation, DER methods project a computed solution onto a constraint set that ensures that the DLO's length does not change. However, this projection operation can introduce instabilities during simulation because it does not ensure that momentum is conserved during motion. This papers adopts a constraint solver based on a PBD model [22], which is described in Section IV-C, to enforce inextensiblity while conserving momentum. This improves the accuracy of prediction when compared to classical methods.

IV. Methodology

This section introduces DDER, which is a reformulation of DER to ensure differentiability. This is a crucial contribution as it enables material properties to be learned from data to reduce the real-to-sim gap. This section then describes how to incorporate a learning-assisted integration method, which allows us to simulate the behavior of a DLO rapidly. Next, this section describes a position-based constraint that allows us to enforce inextensibility while preserving numerical stability during simulation. This results in a prediction algorithm that is described in Algorithm 1. Note that this algorithm can be run over multiple steps to generate a prediction of the state of the DLO over a long time horizon.

A. Differentiable Discrete Elastic Rods (DDER)

We first develop a reformulation of DER such that the DER model is differentiable with respect to the material property parameters. We call this extension Differentiable Discrete Elastic Rods (DDER). This reformulation enables auto-tuning the material property parameters of the DDER model of a DLO using real-world data. To do this, we implement DDER in PyTorch [23]. We then train a learning model to identify the material parameters by minimizing a prediction loss between experimentally collected ground truth data $\mathbf{X}_{t+1}$ and our model prediction $\hat{\mathbf{X}}_{t+1}$ for each time t. Unfortunately, simulating the DDER model to predict $\hat{\mathbf{X}}_{t+1}$ using (2) and (3) requires solving an optimization problem (4). As a result, to minimize the prediction loss, we solve the following bi-level optimization problem for each time t:

$$\min_{\boldsymbol{\alpha}} \quad \|\mathbf{X}_{t+1} - \hat{\mathbf{X}}_{t+1}(\boldsymbol{\theta}_t^*)\|_2 \tag{5}$$

$$\boldsymbol{\theta}_t^* = \arg\min_{\boldsymbol{\theta}_t} P(\hat{\mathbf{X}}_t, \boldsymbol{\theta}_t, \boldsymbol{\alpha}) \tag{6}$$

Note that $\hat{\mathbf{X}}_{t+1}$ depends on the previously computed $\boldsymbol{\theta}_t^*$ due to (2) and (3). To compute the gradient of this optimization problem with respect to $\boldsymbol{\alpha}$, we use Theseus [24].

B. Integration Method with Residual Learning

To compensate for numerical errors that arise from discrete integration methods and to improve the numerical stability of the algorithm that we use to simulate the DDER model, we incorporate a learning-based framework into the integration method:

$$\hat{\mathbf{V}}_{t+1} = \hat{\mathbf{V}}_t - \Delta_t M^{-1}\Big(\frac{\partial P(\hat{\mathbf{X}}_t, \boldsymbol{\theta}_t(\hat{\mathbf{X}}_t), \boldsymbol{\alpha})}{\partial \mathbf{X}_t} + \\ + \text{DNN}(\hat{\mathbf{X}}_t, \boldsymbol{\alpha})\Big), \tag{7}$$

$$\hat{\mathbf{X}}_{t+1} = \hat{\mathbf{X}}_t + \Delta_t\Big(\hat{\mathbf{V}}_{t+1} + \text{DNN}(\hat{\mathbf{X}}_t, \hat{\mathbf{V}}_t, \boldsymbol{\alpha})\Big) \tag{8}$$

where DNN denotes a deep neural network. **We would like to note that this contribution can be applied to standard DER models as well.** We utilize a structure that is similar to residual learning [25], with one major difference: we define a shortcut connection that is equal to the integration method that is derived from the DDER physics-based modeling. Unlike pure learning methods, the short cut connection grounds the DNN in physical laws while also leveraging data-driven insights. As shown in the ablation study results, such an approach significantly improves modeling performance. As we describe in Section VI, we build our DNN using a Graph Neural Network (GNN) [3]. This is motivated in part by recent work that has illustrated how to successfully apply GNNs to model the spatial relationship between vertices in a DLO [10].

C. Inextensiblity Enforcement with PBD

To enforce inextensibility after simulating the DLO forward for a single time step (8), we enforce an additional constraint on the DLO in a manner similar to the PBD method [22]. First, we define a constraint function between spatially successive vertices at each time step:

$$\text{C}(\hat{\mathbf{x}}_{t+1}^i, \hat{\mathbf{x}}_{t+1}^{i+1}) = |||\hat{\mathbf{x}}_{t+1}^i - \hat{\mathbf{x}}_{t+1}^{i+1}||_2 - ||\bar{\mathbf{e}}_i||_2|, \tag{9}$$

where $\bar{\mathbf{e}}$ denotes the edge between vertices i and $i+1$ when the DLO is undeformed. Note that this constraint is non-zero if the distance between successive vertices of the DLO increases.

The PBD method iteratively modifies the positions of each of the vertex using the correction terms $\Delta\hat{\mathbf{x}}_{t+1}^i$ and $\Delta\hat{\mathbf{x}}_{t+1}^{i+1}$ until the norm of the constraint falls below a certain threshold:

$$\text{C}(\hat{\mathbf{x}}_{t+1}^i + \Delta\hat{\mathbf{x}}_{t+1}^i, \hat{\mathbf{x}}_{t+1}^{i+1} + \Delta\hat{\mathbf{x}}_{t+1}^{i+1}) < \epsilon \tag{10}$$

Algorithm 2 Enforcing Inextensibility with Momentum Preserving PBD

Require: $\hat{\mathbf{X}}_{t+1}$ and $\epsilon > 0$
1: **while** any $\mathrm{C}(\hat{\mathbf{x}}^i_{t+1}, \hat{\mathbf{x}}^{i+1}_{t+1}) > \epsilon$ **do**
2: **for** $i = 2$ **to** $n-1$ **do**
3: $\hat{\mathbf{x}}^i_{t+1} = \hat{\mathbf{x}}^i_{t+1} + \beta^i \Delta\hat{\mathbf{x}}^i_{t+1}$ ▷ (11)
4: **end for**
5: **end while**
6: **return** : $\hat{\mathbf{X}}_{t+1}$ ▷ Updated Vertices

The correction, $\Delta\hat{\mathbf{x}}^i_{t+1}$ and $\Delta\hat{\mathbf{x}}^{i+1}_{t+1}$, are applied iteratively until the constraint falls below some user specified threshold, $\epsilon > 0$. To calculate the correction terms, PBD typically applies a Taylor Approximation of the constraint and sets this equal to zero (i.e. the correction to the vertex locations should be chosen to ensure that the constraint is satisfied). The correction term is then given by:

$$\Delta\hat{\mathbf{x}}^i_{t+1} = C(\hat{\mathbf{x}}^i_{t+1}, \hat{\mathbf{x}}^{i+1}_{t+1}) \frac{\hat{\mathbf{x}}^{i+1}_{t+1} - \hat{\mathbf{x}}^i_{t+1}}{||\hat{\mathbf{x}}^{i+1}_{t+1} - \hat{\mathbf{x}}^i_{t+1}||_2} \tag{11}$$

Notably, this correction to the vertex location does not ensure that the linear and angular momentum is preserved. To ensure that linear and angular moment is conserved after correction, one needs to ensure that the following pair of equations are satisfied:

$$\sum_{i=3}^{n-2} \boldsymbol{M}^i \cdot \Delta\hat{\mathbf{x}}^i_{t+1} = \mathbf{0} \tag{12}$$

$$\sum_{i=3}^{n-2} \mathbf{r}^i \times (\boldsymbol{M}^i \cdot \Delta\hat{\mathbf{x}}^i_{t+1}) = \mathbf{0}, \tag{13}$$

where $\mathbf{r}^i$ is the vector between $\hat{\mathbf{x}}^i_{t+1}$ and a common rotation center. Note in particular that if the mass matrix associated with each vertex is not identical, then the correction proposed in (11) leads to a violation of (12) and (13). To address this issue, we modify the correction proposed in (11) by multiplying it by a $\beta \in \mathbb{R}$ that is inversely correlated with the weight ratio between successive vertices (i.e., let $\beta^i = \frac{\boldsymbol{M}^{i+1}}{(||\boldsymbol{M}^i||_2 + ||\boldsymbol{M}^{i+1}||_2)}$ and $\beta^{i+1} = \frac{\boldsymbol{M}^i}{(||\boldsymbol{M}^i||_2 + ||\boldsymbol{M}^{i+1}||_2)}$). Then, if we let the correction between successive steps be equal to $\beta^i \Delta\hat{\mathbf{x}}^i_{t+1}$, one can ensure that (12) and (13) are satisfied.

Algorithm 2 summarizes our proposed method to enforce inextensibility while preserving momentum. Note that in practice, the while loop in Algorithm 2 typically converges after two iterations for $\epsilon = 0.05$. Once we have the output from Algorithm 2, we update the velocities of the vertices to reflect the new vertex locations (i.e., Line 5 of Algorithm 1).

V. DLO Tracking with Modeling

This section describes how we can use DEFORM to perform robust DLO tracking. Notably, DEFORM enables us to deal with occlusions without requiring a frame-by-frame sensor update of the state of the DLO. In particular, this is possible due to our model accurately predicting the state of the DLO over long time horizons. As a result, we utilize a GMM model because it is independent of time.

A. State Estimation in the Presence of Occlusions

Suppose that we are given access to an RGB-D sensor observing our DLO during manipulation and suppose that we translate these measurements at time step t into a point cloud which we denote by $\mathbf{S}_t = \{s^1_t, s^m_t, \ldots, s^M_t\} \in \mathbb{R}^{\mathrm{M}\times 3}$.

To address the limitations of the algorithms discussed in Section II, we leverage our predictions generated by applying Algorithm 1 as follows. We first filter the point cloud at time t using our predicted data. If any element of the point cloud is beyond some distance from our prediction, $\hat{\mathbf{X}}_t$, then we remove it. Let $\tilde{\mathbf{S}}_t$ denote the remaining points in the point cloud. Next, we detect which of the vertices of the DLO are unoccluded from the RGB-D sensor by checking if their predicted depth is close to their observed depth in the sensor. Suppose the number of vertices that are unoccluded is $\tilde{n}$.

Once this is done, we apply DBSCAN to group $\tilde{\mathbf{S}}_t$ into $\tilde{n}+1$ groups. Next, we associate each of the unoccluded vertices with one of the groups by finding the group to which its predicted location is smallest. We then take each group and perform GMM on the group with a number of mixtures equal to the number of vertices j that were associated with that group. Note, each mixture is associated with an unoccluded vertex of the DLO. The new predicted location of the vertex generated by DEFORM is updated by setting the new vertex location equal to the mean of its associated mixture. If a particular vertex was occluded, then its predicted location is left equal to its output from Algorithm 1.

B. Initial State Estimation under Occlusion

Note that in many instances, it can be difficult to estimate the location of the vertices when the sensor turns on and the DLO is occluded. Unfortunately, Algorithm 1 and the state estimation algorithm from the previous subsection require access to a full initial state. Fortunately, we can address this problem by making the following assumption:

Assumption 3. *When the sensor measurements begin, the DLO is static and is only subject to gravity and/or the manipulators which are holding its ends.*

Under the above assumption, we initialize the DLO using an initial guess that aligns with observed vertices. This initial guess is then forward simulated using Algorithm 1 repeatedly until the DLO reaches a static state. This steady state is then used as the predicted state for all the vertices, including the occluded vertices.

VI. Experiments and Results

This section describes the **real-world** evaluation of DEFORM. This includes *1)* a quantitative comparison of modeling accuracy against state-of-the-art methods *2)* an ablation study testing the effect of parameter auto-tuning and residual learning, *3)* an assessment of computational speed, and *4)*

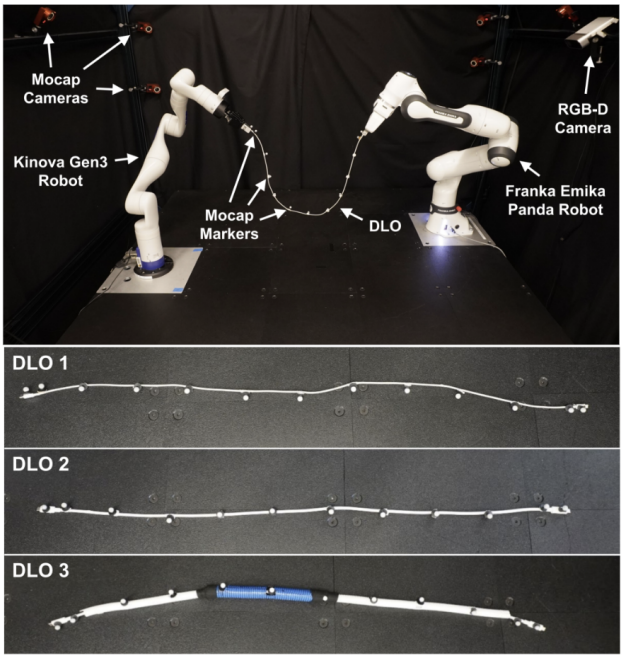


Fig. 3: Top: an illustration of the experimental setup. Bottom: an illustration of the DLOs that are used to evaluate and compare the performance of DEFORM with various state-of-the-art DLO modeling methods.

an illustration of using DEFORM to track DLOs with an RGB-D camera, *5)* a demonstration of using DEFORM for shape matching. Note, all code and data will be provided upon acceptance of the final paper.

A. Experiments Setup

1) Hardware Parameters: We use an OptiTrack motion capture system to obtain the ground truth vertex locations for DLOs as depicted in Figure 3. Spherical markers with a diameter of 7.9 mm and weight of 0.4 g, are attached to the DLOs using the OptiTrack tracking system. Ten Flex3 cameras capture the motion of markers at a frequency of 100Hz, with a positional error of less than 0.3mm. Note that the markers are used to create training and evaluation datasets, as well as evaluate the performance of DEFORM by providing the ground truth. A Franka Emika Panda robot arm and a Kinova Gen3 robot arm are utilized for dual manipulation. For perception with an RGB-D camera, we use an Azure Kinect DK with a resolution of 720 x 1080 and a frequency of 30 Hz. Three distinct cables were constructed to validate the modeling accuracy of DEFORM during observation and manipulation tasks. The physical properties of each wire are outlined in Table I. These properties include the length, weight, and stiffness of each DLO, as well as the number of Mocap markers attached to it. The stiffness of each DLO is ranked on a relative scale. Note, we set the number of vertices equal to the number of Mocap markers.

2) Dataset Collection: For each of the three DLOs, we collect 350 seconds of dynamic trajectory data in the real-world using the motion capture system at a perception frequency of 100 Hz.

TABLE I: A summary of the material properties of each of the wires that were observed/manipulated during the real-world experiment.

Name	Length [m]	Weight [g]	Stiffness	# Mocap Markers
DLO 1	1.152	34.5	3	13
DLO 2	0.996	65.2	4	12
DLO 3	0.998	96.8	5	12

TABLE II: Baseline comparison results for modeling real-world DLO.

Modeling (L1)	DLO 1	DLO 2	DLO 3
XPBD[8]	0.0400	0.0385	0.0380
DRM[18]	0.0364	0.0416	0.0321
Bi-LSTM[4]	0.0198	0.0175	0.0110
GNN[10]	0.0341	0.0223	0.0215
DER[7]	0.0196	0.0191	0.0186
DEFORM	**0.0101**	**0.0097**	**0.0077**

B. Implemenation Details of DEFORM

We implement DEFORM with PyTorch and use the Levenberg-Marquardt algorithm as the solver for Theseus. We utilize PyTorch for training and Numpy for non-batched prediction. We initialize the length and mass parameters of the DLO according to Table I. The other material properties are initialized randomly and learned during the training process.

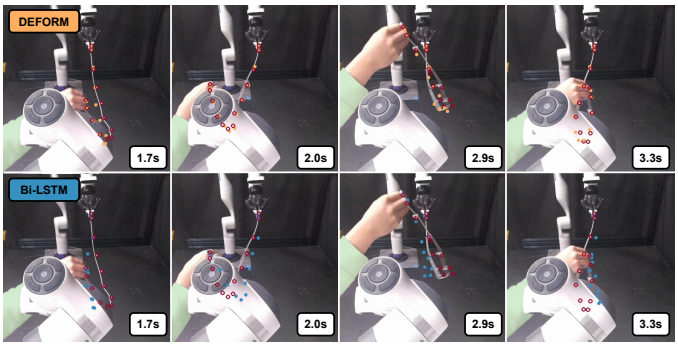
As discussed in Section.IV-B, DEFORM uses a GNN to perform residual learning during the integration step. We adapt the default implementation of a graph convolution network (GCN) from PyTorch and set its feature dimension to be 32. $\hat{\mathbf{X}}_t$ and $\hat{\mathbf{V}}$ are feature-wise concatenated as $(\hat{\mathbf{X}}_t, \hat{\mathbf{V}}) \in \mathbb{R}^{n\times 6}$ and are the input of the GCN. The outputs of the GCN are flattened and decoded by a MLP constructed with two linear layers with a Rectified Linear Unit (ReLU) in the middle.

C. Baseline Comparisons

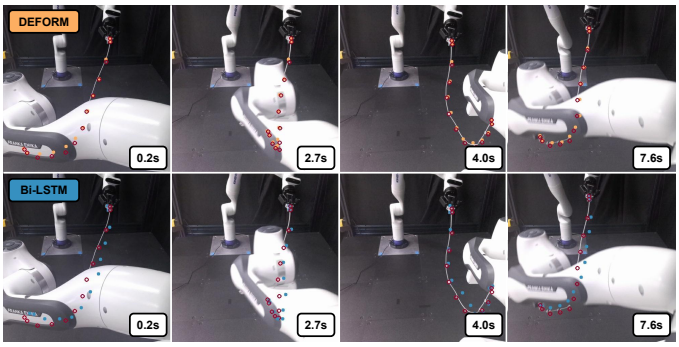
We compare our approach to a variety of methods using either physics-based[7], [8], [18] or learning-based models[4], [10]. The model for each of the three DLOs is trained separately using the trajectory data collected from our real-world experiments. In each training iteration, models are trained using the same 100 steps (1s) of dynamic motion. This includes a variety of different behaviors including slow and fast motions. The dataset is split with 80% for training and 20% for evaluation. The L1 loss over the 100 step prediction horizon is used for training and a 500 step prediction horizon is used for evaluation of all models. Results are outlined in Table II.

D. Modeling Results

To begin, Table II summarizes the results of using DEFORM when compared to a variety of baselines using the L1 loss. Each method's performance is evaluated when predicting the state of the wire over a 5s prediction horizon. Note that DEFORM outperforms all baselines for each of the five evaluated cables.



(a) DLO 1, robot-human manipulation



(b) DLO 1, dual-robot manipulation

Fig. 4: Visualization of the performance of DEFORM and Bi-LSTM in predicting trajectories for DLO 1. The ground truth vertices of the DLOs are marked with red hollow circles. The predicted vertices are marked with orange solid circles for DEFORM and blue solid circles for the Bi-LSTM model, respectively.

TABLE III: Results of the ablation study.

Methods	DLO 1	DLO 2	DLO 3
DER	0.0196	0.0191	0.0186
W/O Residual Learning	0.0154	0.0177	0.0142
W/O Auto-Parameter Tuning	0.0121	0.0132	0.0109
DEFORM	**0.0101**	**0.0097**	**0.0077**

TABLE IV: Computational time of one step inference [s]

Methods	DLO 1	DLO 2	DLO 3
XPBD[8]	0.0145	0.0138	0.0133
DRM[18]	0.00453	0.00440	0.00431
Bi-LSTM[4]	**0.00036**	**0.00033**	**0.00034**
GNN[10]	0.0169	0.00167	0.00159
DER[7]	0.0181	0.0177	0.0176
DEFORM	0.00967	0.00922	0.0094

1) Ablation Study: Next, we summarize an ablation study that was run to demonstrate the efficacy of our design choice. The results are summarized in Table V where a comparison to DER is included. Note, DER does not include residual learning or rely upon auto-tuning the material property parameters as was described in Section IV-A. The results of the study show that residual learning has the biggest on the performance improvement, but the auto-tuning of material properties has a non-trivial affect as well.

2) Computational Speed: Next, we compare computational speed for one step inference. Table IV indicates that Bi-LSTM has the fastest speed. DEFORM is the third fastest and is able to generate single step predictions at a 100Hz frequency.

E. State Estimation Results

We next evaluate the performance of DEFORM when it is used in concert with sensor observations for state estimation. These experiments included occlusions of the DLO that arose from the manipulators or humans moving in front of the sensor. In this experiment, we compare how well the state estimation framework described in Section V does when using the DEFORM model versus when using the Bi-LSTM model. Bi-LSTM was chosen since it was the second best model in terms of model accuracy. We also compare how well each of these methods do with varying frequencies of sensor measurement updates. As in the earlier experiment, we evaluate the L1 norm of the prediction over a 5s long prediction horizon.

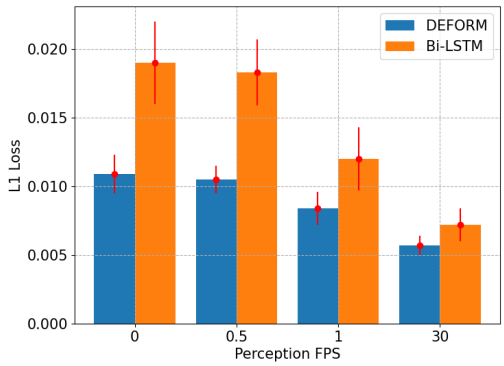


Fig. 5: The L1 loss while performing state estimation.

An illustration of the performance of these methods can be found in Figure 4. The results of this experiment can be found in Figure 5. Unsurprisingly, each of the methods improve when they are given access to more frequent sensor updates. This illustrates that the state estimation algorithm described in Section V behaves appropriately.

F. 3-D Shape Matching Manipulation

To demonstrate the practical applicability of DEFORM, we illustrate its use for DLO 3-D shape matching tasks in both simulation and the real world

To do this, we rely upon ARMOUR[26], an optimization-based motion planning and control framework. More details about ARMOUR's trajectory parameterization and associated closed loop controller can be found in [26, Section IX]. The cost function minimizes the distance between the predicted DLO configuration given a chosen robot trajectory and a target DLO configuration. The predicted state of the DLO is

TABLE V: Shape matching manipulation success rate

Methods	Real-World	Simulation
Bi-LSTM	10/20	78/100
DEFORM	**17/20**	**90/100**

computed via a DLO modeling technique.

In this evaluation, we compare the performance DEFORM and Bi-LSTM when each is used to predict the state of the DLO while following the trajectory chosen by ARMOUR. We compare the performance of each method on 100 random cases in a PyBullet simulation and 20 random cases in the real world. The manipulation algorithm is given 30 seconds to complete the task. The process is deemed successful if the Euclidean distance between each individual pair of planned and target DLO vertices is less than 0.05 meters. The results, summarized in Table V, illustrate that the DEFORM model achieves a higher success rate in both simulated and real-world experiments when compared to Bi-LSTM.

VII. Conclusion

This paper presents DEFORM, which is a method to model and predict the behavior of DLOs accurately over long time horizon under dynamic manipulation. To demonstrate DEFORM's efficacy, this paper performs a comprehensive set of experiments that evaluate accuracy, computational speed, and its usefulness for manipulation and perception tasks3.3. When compared to the state-of-the-art, DEFORM is more accurate while being sample efficient without sacrificing computational speed. This paper also illustrates the practicality of applying DEFORM for state estimation to track a DLO during dynamic manipulation in environments with occlusions.

References

[1] Y. Adagolodjo, L. Goffin, M. De Mathelin, and H. Courtecuisse, "Robotic insertion of flexible needle in deformable structures using inverse finite-element simulation," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 697–708, 2019.

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

[3] "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81,

[4] M. Yan, Y. Zhu, N. Jin, and J. Bohg, *Self-supervised learning of state estimation for manipulating deformable linear objects*, 2020.

[5] M. García, C. Mendoza, L. Pastor, and A. Rodríguez, "Optimized linear fem for modeling deformable objects," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 393–402, 2006.

[6] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98, New York, NY, USA: Association for Computing Machinery, 1998, 43–54.

[7] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," in *ACM SIGGRAPH 2008 Papers*, ser. SIGGRAPH '08, Los Angeles, California: Association for Computing Machinery, 2008.

[8] F. Liu, E. Su, J. Lu, M. Li, and M. C. Yip, *Differentiable robotic manipulation of deformable rope-like objects using compliant position-based dynamics*, 2022.

[9] A. Sintov, S. Macenski, A. Borum, and T. Bretl, "Motion planning for dual-arm manipulation of elastic rods," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Jul. 2020.

[10] C. Wang, Y. Zhang, X. Zhang, *et al.*, "Offline-online learning of deformation model for cable manipulation with graph neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, 5544–5551, Apr. 2022.

[11] "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378,

[12] J. Liang, M. Lin, and V. Koltun, "Differentiable cloth simulation for inverse problems," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019.

[13] A. Myronenko, X. Song, and M. Carreira-Perpinan, "Non-rigid point set registration: Coherent point drift," *Advances in neural information processing systems*, vol. 19, 2006.

[14] C. Chi and D. Berenson, "Occlusion-robust deformable object tracking without physics simulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Nov. 2019.

[15] Y. Wang, D. McConachie, and D. Berenson, *Tracking partially-occluded deformable objects while enforcing geometric constraints*, 2020.

[16] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Iterative residual policy for goal-conditioned dynamic manipulation of deformable objects," in *Proceedings of Robotics: Science and Systems (RSS)*, 2022.

[17] D. Berenson, "Manipulation of deformable objects without modeling and simulating deformation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4525–4532.

[18] M. Ruan, D. McConachie, and D. Berenson, "Accounting for directional rigidity and constraints in control for manipulation of deformable objects without physical simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 512–519.

[19] S. Dharmavaram, H. Ebrahimi, and R. Ghosh, "Coupled bend–twist mechanics of biomimetic scale substrate," *Journal of the Mechanics and Physics of Solids*, vol. 159, p. 104 711, 2022.

[20] F. M. Filotto, F. Runkel, and G. Kress, "Cross section shape optimization of wire strands subjected to purely tensile loads using a reduced helical model," *Adv. Model. Simul. Eng. Sci.*, vol. 7, no. 1, p. 23, 2020.

[21] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun, "Efficient simulation of inextensible cloth," *ACM Trans. Graph.*, vol. 26, no. 3, 49–es, 2007.

[22] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *J. Vis. Comun. Image Represent.*, vol. 18, no. 2, 109–118, 2007.

[23] A. Paszke, S. Gross, F. Massa, *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019.

[24] L. Pineda, T. Fan, M. Monge, *et al.*, *Theseus: A library for differentiable nonlinear optimization*, 2023.

[25] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015.

[26] J. Michaux, P. Holmes, B. Zhang, *et al.*, *Can't touch this: Real-time, safe motion planning and control for manipulators under uncertainty*, 2023.

VIII. Contribution Clarification

This project is initially proposed by my research advisor Prof. Ram Vasudevan and I have worked on this project across four semsters.

Contributions to research: developed and implemented the first four primary contributions of DEFORM. Collected the DLO1 and DLO2 datasets. Implemented, analyzed and finalized all modeling and perception-related experiments. Assisted co-authors in the motion planning experiment by providing the DEFORM code, adjusting the cost function, and debugging the code. Contributions to paper writing: wrote the initial draft of the paper and assisted co-authors in revising it. Provided data for all visualizations included in the paper.