

## A Algorithmic Details

### A.1 DNN Structure

We built our DNN using a Graph Neural Network (GNN) [5]. 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 [15]. We adapt the default implementation of a graph convolution network (GCN) [39] found in the PyTorch Geometric library [40]. GCNs have demonstrated their ability to reduce computational costs while effectively extracting informative latent representations from graph-structured data.  $\hat{\mathbf{X}}_t$  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. We set the feature dimension to 32 for message passing, allowing each node to receive information from its local neighborhood. We aggregate each node’s neighbors’ features using summation. 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.

### A.2 Summary and Discussion of Improved Inextensibility Enforcement with PBD

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). Further, as shown in Table 6, skipping Algorithm 2 in Algorithm 1 and relying solely on DNNs to capture the inextensibility of DLOs can lead to simulation instability. This instability arises because modeling stiff behavior, such as the inextensibility of DLOs, makes the learning process highly sensitive to inputs and hinders effective gradient propagation [8, 11]. On the contrary, our approach explicitly imposes inextensibility, and separates DNNs from the complications associated with learning stiff behavior.

---

#### Algorithm 2 Enforcing Inextensibility with Momentum Preserving PBD

---

**Require:**  $\hat{\mathbf{X}}_{t+1}$  and  $\epsilon > 0$

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

---

### A.3 Training Setup

Let  $\mathbf{U}_{1:T-1}$  denote the set of inputs applied between times  $t = 1$  and  $t = T - 1$ , and let  $\mathbf{X}_{1:T} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T\} \in \mathbb{R}^{T \times n \times 3}$  denote the associated ground truth trajectory of the DLO from times  $t = 1$  to  $t = T$ . With known  $\mathbf{X}_1$ ,  $\mathbf{V}_1$  and  $\mathbf{U}_{1:T-1}$ , Algorithm 1 can be applied recursively to generate predicted associated trajectory  $\hat{\mathbf{X}}_{2:T}$ . Let  $\phi$  denote parameters of DNN. The objective of training is to solve the following optimization problem:

$$\min_{\alpha, \phi} \sum_{t=1}^{T-1} \|\mathbf{X}_{t+1} - \hat{\mathbf{X}}_{t+1}\|_2 \quad (13)$$

By taking advantage of DEFORM’s differentiability,  $T$  can be set to values greater than 2 to capture long-term behavior. This multi-step training pipeline results in higher prediction accuracy than the single-step training pipeline, as shown in Table 6.

## B Experimental Details

### B.1 Hardware Parameters

We use an OptiTrack motion capture (Mocap) system to obtain the ground truth vertex locations for DLOs as depicted in Figure 6. 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

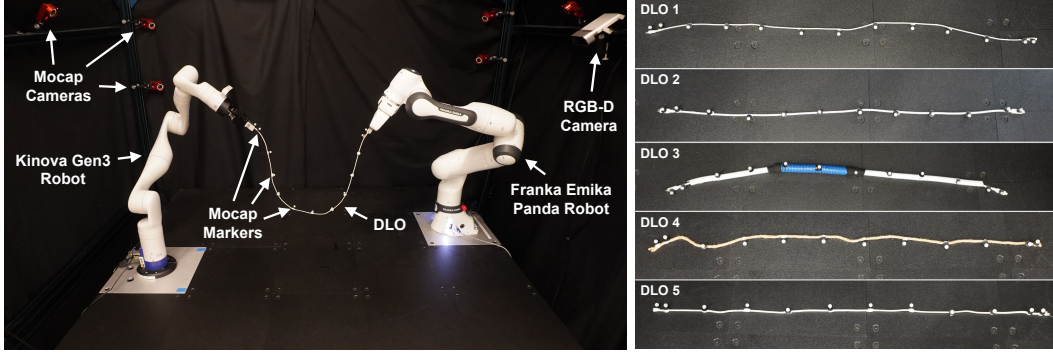


Figure 6: **Left:** An illustration of the experimental setup. **Right:** An illustration of the DLOs that are used to evaluate and compare the performance of DEFORM with various state-of-the-art DLO modeling methods.

the markers at a frequency of 100Hz, with a positional error of less than 0.3mm. 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 and two distinct ropes were constructed as shown in Figure 6. The physical properties of each wire are outlined in Table 4. 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.

Table 4: Material Properties

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
DLO 4	0.973	22.0	1	12
DLO 5	0.988	19.2	2	12

## B.2 Software Implementation:

All experiments were conducted in Python, on an Ubuntu 20.04 machine equipped with an AMD Ryzen PRO 5995WX CPU, 256GB RAM, and 128 cores. All training pipelines were built within the PyTorch training framework. 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 4. The other material properties are initialized randomly and learned during the training process. We use SGD optimizer with  $10^{-4}$  learning rate for training.

## B.3 Ablation Study: DLO4 and DLO5

As a supplement for Table 2, the ablation study of DLO4 and DLO5 is shown in Table 5.

## B.4 Ablation Study: Inextensibility Enforcement with Learning and Single-step Training

We conducted a further ablation study on training DEFORM without the improved inextensibility enforcement and training DEFORM using only a single-step prediction as shown in Table 6. When DEFORM is trained without enforcing inextensibility, simulation instability results in very high prediction loss, highlighting the importance of properly enforcing inextensibility over long time horizons. Additionally, training DEFORM using only a single-step prediction results in lower long-term prediction accuracy compared to training DEFORM using a 100-step prediction, demonstrating the importance of DEFORM’s differentiability for accurate long-term predictions.

Table 5: Ablation Study with DLO 4 and DLO 5.

Method	Accuracy ( $10^{-2}\text{m}$ )	
	4	5
DER	1.50	1.65
W/O Residual Learning	1.33	1.26
W/O System ID	1.02	1.10
Original Inextensibility	1.29	1.58
DEFORM	<b>0.850</b>	<b>0.987</b>

Table 6: Additional Ablation Study.

Method	Accuracy ( $10^{-2}\text{m}$ )		
	1	2	3
W/O Enforcing Inextensibility with PBD	$7.7 \times 10^5$	$260 \times 10^5$	$310 \times 10^5$
Single Step Prediction Training	1.82	1.74	1.79
DEFORM	<b>1.01</b>	<b>0.97</b>	<b>0.77</b>

## B.5 ARMOUR

To perform a shape matching task, we rely upon ARMOUR[41], an optimization-based motion planning and control framework. The goal of the shape matching task is to use a robot arm to manipulate the DLO from an initial configuration to a predefined target configuration. To accomplish this goal, ARMOUR performs planning in a receding horizon fashion. During each planning iteration, ARMOUR selects a trajectory to follow by solving an optimization problem. More details about ARMOUR’s trajectory parameterization and associated closed loop controller can be found in [41, 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 computed via a DLO modeling technique.

## C DLO Tracking with Modeling

This section first discusses the difficulties of incorporating existing framework with modeling for long-time DLO tracking under occlusion. It then proposes a novel perception pipeline, which is utilized in Section 5.2.

### C.1 DLO Tracking Review

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 [42, 43, 44, 45, 46, 47, 48, 49]. 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[45, 46] 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. [45, 46, 47, 48, 49] Though powerful, to work accurately, these methods require frequent measurement updates and can struggle in the presence of occlusions for long time horizons. Recent research has explored particle filtering within a lower-dimensional latent space embedding and applied learning-based techniques for shape estimation under occlusion [50, 51]. Each of these methods rely upon different models for DLOs that tend to have numerical instabilities when used for prediction. As a result, these perception methods require high frequency sensor measurement updates to behave accurately. This paper illustrates that our

---

**Algorithm 3** State Estimation with Presence of Occlusion

---

**Inputs:**  $\hat{\mathbf{X}}_{t+1}$  ▷ Algorithm. 1

- 1:  $\mathbf{S}_{T+1} \leftarrow$  RGB-D Camera
- 2:  $\tilde{\mathbf{S}}_{T+1} \leftarrow \text{filter}(\hat{\mathbf{X}}_{t+1}, \mathbf{S}_{T+1})$
- 3: Unoccluded  $\hat{\mathbf{X}}_{t+1} \leftarrow \text{Depth Matching}(\tilde{\mathbf{S}}_{T+1}, \hat{\mathbf{X}}_{t+1})$
- 4:  $n + 1$  groups  $\leftarrow \text{DBSCAN}(\tilde{\mathbf{S}}_{T+1})$
- 5: **for** each group **do**
- 6:     GMM Number  $j \leftarrow \text{Match}(\text{Unoccluded } \hat{\mathbf{X}}_{t+1}, \text{group})$
- 7:     Mixture Center  $\leftarrow \text{GMM}(\text{group}, j)$
- 8: **end for**
- 9: **for** each  $\hat{\mathbf{X}}_{t+1}$  **do**
- 10:    **if** Observed **then**
- 11:      Vertex  $\leftarrow$  Associated Mixture Center
- 12:    **else**
- 13:      Vertex  $\leftarrow$  Predicted Vertex
- 14:    **end if**
- 15: **end for**
- 16: **return** : Vertices

---

---

**Algorithm 4** Tracking DLO with Modeling

---

**Initial State Estimation**

**Require:**  $\mathbf{S}_1 \leftarrow$  RGB-D Camera

- 1: DLO Initial Guess  $\leftarrow \mathbf{S}_1$
- 2: **while** DLO not Static **do**
- 3:     Execute DEFORM ▷ Algorithm. 1
- 4: **end while**
- 5: Return  $\hat{\mathbf{X}}_1, \hat{\mathbf{V}}_1 = \mathbf{0}$

**Tracking DLO under Manipulation**

- 1: **while** Tracking DLO **do**
- 2:    **while** Predicting DLO **do**
- 3:      Execute DEFORM ▷ Algorithm. 1
- 4:    **end while**
- 5:    State Estimation and Correction
- 6: **end while**

---

540 proposed model allows us to adapt DEFORM’s long time horizon prediction capability to relax the  
541 frequency of sensor updates, which reduces the overall computational cost of tracking DLOs in the  
542 presence of occlusions.

## 543 C.2 DLO Tracking with DEFORM

544 This section describes how we can use DEFORM to perform robust DLO tracking. Notably, DE-  
545 FORM enables us to deal with occlusions without requiring a frame-by-frame sensor update of the  
546 state of the DLO. In particular, this is possible due to our model accurately predicting the state of  
547 the DLO over long time horizons. As a result, we utilize a GMM model because it is independent  
548 of time. The state estimation approach is outlined in Algorithm 3. Additionally, Algorithm 4 sum-  
549 marizes the perception pipeline that describes the tracking of DLOs with state estimation and initial  
550 state estimation using DEFORM.

## 551 C.3 State Estimation in the Presence of Occlusions

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

555 To address the limitations of the algorithms discussed in Section 2, we leverage our predictions  
 556 generated by applying Algorithm 1 as follows. We first filter the point cloud at time  $t$  using our  
 557 predicted data. If any element of the point cloud is beyond some distance from our prediction,  $\hat{\mathbf{X}}_t$ ,  
 558 then we remove it. Let  $\tilde{\mathbf{S}}_t$  denote the remaining points in the point cloud. Next, we detect which of  
 559 the vertices of the DLO are unoccluded from the RGB-D sensor by checking if their predicted depth  
 560 is close to their observed depth in the sensor. Suppose the number of vertices that are unoccluded is  
 561  $\tilde{n}$ .

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

#### 569 **C.4 Initial State Estimation under Occlusion**

570 Note that in many instances, it can be difficult to estimate the location of the vertices when the sensor  
 571 turns on and the DLO is occluded. Unfortunately, Algorithm 1 and the state estimation algorithm  
 572 from the previous subsection require access to a full initial state. Fortunately, we can address this  
 573 problem by making the following assumption: When the sensor measurements begin, the DLO is  
 574 static and is only subject to gravity and/or the manipulators which are holding its ends. Under the  
 575 above assumption, we initialize the DLO using an initial guess that aligns with observed vertices.  
 576 This initial guess is then forward simulated using Algorithm 1 repeatedly until the DLO reaches a  
 577 static state. This steady state is then used as the predicted state for all the vertices, including the  
 578 occluded vertices.