

442 **A Algorithmic Details**

443 **A.1 DNN Structure**

444 We built our DNN using a Graph Neural Network (GNN) [5]. This is motivated in part by re-
 445 cent work that has illustrated how to successfully apply GNNs to model the spatial relationship
 446 between vertices in a DLO [15]. We adapt the default implementation of a graph convolution net-
 447 work (GCN) [39] found in the PyTorch Geometric library [40]. GCNs have demonstrated their
 448 ability to reduce computational costs while effectively extracting informative latent representations
 449 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
 450 the input of the GCN. We set the feature dimension to 32 for message passing, allowing each node
 451 to receive information from its local neighborhood. We aggregate each node’s neighbors’ features
 452 using summation. The outputs of the GCN are flattened and decoded by a MLP constructed with
 453 two linear layers with a Rectified Linear Unit (ReLU) in the middle.

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

455 Algorithm 2 summarizes our proposed method
 456 to enforce inextensibility while preserving mo-
 457 mentum. Note that in practice, the while loop
 458 in Algorithm 2 typically converges after two it-
 459 erations for $\epsilon = 0.05$. Once we have the output
 460 from Algorithm 2, we update the velocities of
 461 the vertices to reflect the new vertex locations
 462 (i.e., Line 5 of Algorithm 1). Further, as shown
 463 in Table 6, skipping Algorithm 2 in Algorithm
 464 1 and relying solely on DNNs to capture the in-
 465 extensibility of DLOs can lead to simulation instability. This instability arises because modeling
 466 stiff behavior, such as the inextensibility of DLOs, makes the learning process highly sensitive to
 467 inputs and hinders effective gradient propagation [8, 11]. On the contrary, our approach explicitly
 468 imposes inextensibility, and separates DNNs from the complications associated with learning stiff
 469 behavior.

Algorithm 2 Enforcing Inextensibility with Mo-
 mentum 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: $\hat{\mathbf{x}}_{t+1}^i = \hat{\mathbf{x}}_{t+1}^i + \beta^i \Delta \hat{\mathbf{x}}_{t+1}^i$ ▷ (11)
 4: **end for**
 5: **end while**
 6: **return** : $\hat{\mathbf{X}}_{t+1}$ ▷ Updated Vertices

470 **A.3 Training Setup**

471 Let $\mathbf{U}_{1:T-1}$ denote the set of inputs applied between times $t = 1$ and $t = T - 1$, and let $\mathbf{X}_{1:T} =$
 472 $\{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_T\} \in \mathbb{R}^{T \times n \times 3}$ denote the associated ground truth trajectory of the DLO from times
 473 $t = 1$ 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
 474 predicted associated trajectory $\hat{\mathbf{X}}_{2:T}$. Let ϕ denote parameters of DNN. The objective of training is
 475 to solve the following optimization problem:

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

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

479 **B Experimental Details**

480 **B.1 Hardware Parameters**

481 We use an OptiTrack motion capture (Mocap) system to obtain the ground truth vertex locations for
 482 DLOs as depicted in Figure 6. Spherical markers with a diameter of 7.9 mm and weight of 0.4 g, are
 483 attached to the DLOs using the OptiTrack tracking system. Ten Flex3 cameras capture the motion of

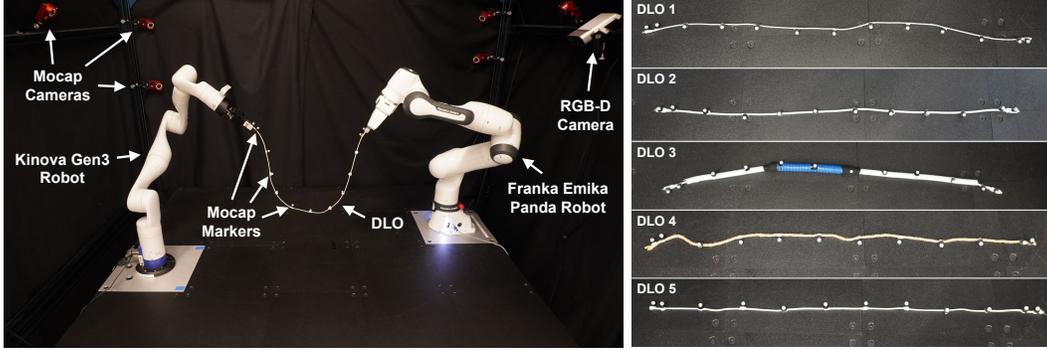


Figure 6: **Left:** An illustration of the experimental setup. **Right:** An illustration of the DLOs that are used to evaluate and compare the performance of DEFORM with various state-of-the-art DLO modeling methods.

484 the markers at a frequency of 100Hz, with a positional error of less than 0.3mm. For perception with
 485 an RGB-D camera, we use an Azure Kinect DK with a resolution of 720 x 1080 and a frequency
 486 of 30 Hz. Three distinct cables and two distinct ropes were constructed as shown in Figure 6. The
 487 physical properties of each wire are outlined in Table 4. These properties include the length, weight,
 488 and stiffness of each DLO, as well as the number of Mocap markers attached to it. The stiffness of
 489 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

490 B.2 Software Implementation:

491 All experiments were conducted in Python, on an Ubuntu 20.04 machine equipped with an AMD
 492 Ryzen PRO 5995WX CPU, 256GB RAM, and 128 cores. All training pipelines were built within
 493 the PyTorch training framework. We implement DEFORM with PyTorch and use the Levenberg-
 494 Marquardt algorithm as the solver for Theseus. We utilize PyTorch for training and Numpy for non-
 495 batched prediction. We initialize the length and mass parameters of the DLO according to Table 4.
 496 The other material properties are initialized randomly and learned during the training process. We
 497 use SGD optimizer with 10^{-4} learning rate for training.

498 B.3 Ablation Study: DLO4 and DLO5

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

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

501 We conducted a further ablation study on training DEFORM without the improved inextensibility
 502 enforcement and training DEFORM using only a single-step prediction as shown in Table 6. When
 503 DEFORM is trained without enforcing inextensibility, simulation instability results in very high
 504 prediction loss, highlighting the importance of properly enforcing inextensibility over long time
 505 horizons. Additionally, training DEFORM using only a single-step prediction results in lower long-
 506 term prediction accuracy compared to training DEFORM using a 100-step prediction, demonstrating
 507 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} 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	0.850	0.987

Table 6: Additional Ablation Study.

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

508 B.5 ARMOUR

509 To perform a shape matching task, we rely upon ARMOUR[41], an optimization-based motion
510 planning and control framework. The goal of the shape matching task is to use a robot arm to ma-
511 nipulate the DLO from an initial configuration to a predefined target configuration. To accomplish
512 this goal, ARMOUR performs planning in a receding horizon fashion. During each planning iter-
513 ation, ARMOUR selects a trajectory to follow by solving an optimization problem. More details
514 about ARMOUR’s trajectory parameterization and associated closed loop controller can be found in
515 [41, Section IX]. The cost function minimizes the distance between the predicted DLO configura-
516 tion given a chosen robot trajectory and a target DLO configuration. The predicted state of the DLO is
517 computed via a DLO modeling technique.

518 C DLO Tracking with Modeling

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

522 C.1 DLO Tracking Review

523 If a DLO is fully observable, then current state-of-the-art methods estimate the location of the DLO’s
524 vertices by applying a Gaussian Mixture Model (GMM), performing clustering, and then using
525 Expectation-Maximization [42, 43, 44, 45, 46, 47, 48, 49]. The output of this GMM algorithm is
526 the mean locations in \mathbb{R}^3 of each of the Gaussians, which is then set equal to the vertex locations
527 of the DLO. However, in practical applications, occlusion of DLOs during manipulation often leads
528 to perception challenges, which complicates accurate prediction. In particular, due to occlusions,
529 one cannot simply set the number of mixtures in the GMM equal to the number of vertices of the
530 DLO. Doing so results in the vertices being incorrectly distributed only to the unoccluded parts of
531 the DLO. Some of the aforementioned methods[45, 46] have been applied to perform tracking of
532 DLOs under occlusion. Typically, this is done by leveraging short time horizon prediction with
533 geometric regularization using prior observations. [45, 46, 47, 48, 49] Though powerful, to work
534 accurately, these methods require frequent measurement updates and can struggle in the presence
535 of occlusions for long time horizons. Recent research has explored particle filtering within a lower-
536 dimensional latent space embedding and applied learning-based techniques for shape estimation
537 under occlusion [50, 51]. Each of these methods rely upon different models for DLOs that tend to
538 have numerical instabilities when used for prediction. As a result, these perception methods require
539 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: $\tilde{\mathbf{S}}_{T+1} \leftarrow$ filter($\hat{\mathbf{X}}_{t+1}, \mathbf{S}_{T+1}$)
- 3: Unoccluded $\hat{\mathbf{X}}_{t+1} \leftarrow$ Depth Matching($\tilde{\mathbf{S}}_{T+1}, \hat{\mathbf{X}}_{t+1}$)
- 4: $n + 1$ groups \leftarrow DBSCAN($\tilde{\mathbf{S}}_{T+1}$)
- 5: **for** each group **do**
- 6: GMM Number $j \leftarrow$ Match(Unoccluded $\hat{\mathbf{X}}_{t+1}$, group)
- 7: Mixture Center \leftarrow GMM(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** ▷ Algorithm. 1
- 3: Execute DEFORM
- 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** ▷ Algorithm. 1
- 3: Execute DEFORM
- 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, $\tilde{\mathbf{X}}_t$,
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.