TrackDLO: Tracking Deformable Linear Objects Under Occlusion with Motion Coherence

Jingyi Xiang[1], Holly Dinkel[2], Harry Zhao[3], Naixiang Gao[1], Brian Coltin[4], Trey Smith[4], Timothy Bretl[2]

***Abstract*—The TrackDLO algorithm estimates the shape of a Deformable Linear Object (DLO) under occlusion from a sequence of RGB-D images. TrackDLO is vision-only and runs in real-time. It requires no external state information from physics modeling, simulation, visual markers, or contact as input. The algorithm improves on previous approaches by addressing three common scenarios which cause tracking failure: tip occlusion, mid-section occlusion, and self-occlusion. This is achieved through the application of Motion Coherence Theory to impute the spatial velocity of occluded nodes, the use of the topological geodesic distance to track self-occluding DLOs, and the introduction of a non-Gaussian kernel that only penalizes lower-order spatial displacement derivatives to reflect DLO physics. Improved real-time DLO tracking under mid-section occlusion, tip occlusion, and self-occlusion is demonstrated experimentally. The source code and demonstration data are publicly released.**

***Index Terms*—RGB-D Perception, Visual Tracking, Perception for Grasping and Manipulation.**

I. Introduction

THIS work presents TrackDLO, an algorithm for real-time tracking of Deformable Linear Object (DLO) (e.g., rope, wire, tubing) shapes. The TrackDLO algorithm tracks DLOs in RGB-D imagery for manipulation tasks, such as knot tying or wire routing, or to monitor DLOs for collision prevention [1]–[5]. These tasks are common in applications including robotic surgery, industrial automation, power line avoidance, and human habitat maintenance [6]–[9]. Several methods track DLOs in real-time with and without visual markers and physics simulation [10]–[15]. The TrackDLO algorithm builds on existing tracking work by addressing three scenarios common in manipulation tasks which cause tracking failure: tip occlusion, mid-section occlusion, and self-occlusion. This work makes the following listed contributions:

1) TrackDLO accurately tracks DLOs in real-time with tip occlusion without information from modeling, simulation, visual markers, or contact. This is achieved by preserving the total length of the DLO and using Motion Coherence Theory (MCT) to impute the spatial velocity of occluded tip nodes from the spatial velocity of visible nodes [16]. The spatial velocity is used to update the DLO shape estimate.
2) TrackDLO tracks self-occluding DLOs without modeling entanglement. This is achieved by incorporating the geodesic distance into the kernel describing how pairs of nodes influence each other's motion.
3) TrackDLO reflects the physics of DLOs by introducing a kernel which only penalizes lower-order spatial velocity derivatives. This work analytically derives forms the kernel can take to satisfy optimality in the Expectation-Maximization algorithm.
4) Data and source code for DLO tracking are released at: https://github.com/RMDLO/trackdlo. A supplementary video demonstrating tracking performance on two DLOs with different material properties is available online.

Manuscript received: March 30, 2023; Revised: June 24, 2023; Accepted: July 20, 2023.

This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments.

[1]Jingyi Xiang and Naixiang Gao are with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. e-mail: {jingyix4, ngao4}@illinois.edu

[2]Holly Dinkel and Timothy Bretl are with the Department of Aerospace Engineering and Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. e-mail: {hdinkel2, tbretl}@illinois.edu

[3]Harry Zhao is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305 USA. e-mail: harzhao@stanford.edu

[4]Brian Coltin and Trey Smith are with the Intelligent Robotics Group, NASA Ames Research Center, Moffett Field, CA, 94035 USA. e-mail: {brian.coltin, trey.smith}@nasa.gov.

Digital Object Identifier (DOI): see top of this page.

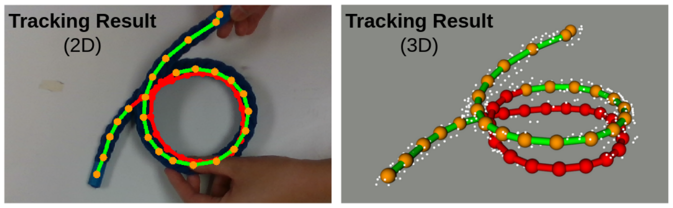


Fig. 1. TrackDLO performs occlusion-robust 3D DLO tracking without physics simulation or other external state information using motion coherence. TrackDLO accurately predicts the location for occluded nodes (red) by imputing their spatial velocities from visible nodes (orange) under mid-section occlusion, tip occlusion, and self-occlusion.

II. Related Work

The Coherent Point Drift (CPD) algorithm performs non-rigid registration to map one set of points onto another. The CPD algorithm uses Gaussian Mixture Model (GMM) clustering and Motion Coherence Theory (MCT) with Expectation-Maximization (EM) to find the probability distribution parameters which maximize the likelihood that a predicted point set corresponds to the original point set [16]–[18]. The Global-Local Topology Preservation (GLTP) algorithm performs modified non-rigid point set registration. The objective function for EM in GLTP uses CPD with locally linear embedding to preserve local topology [19].

Non-rigid point set registration from CPD and GLTP forms the foundation for several algorithms which perform DLO tracking under occlusion. The CPD+Physics algorithm uses CPD for node registration and simulates the DLO in a physics

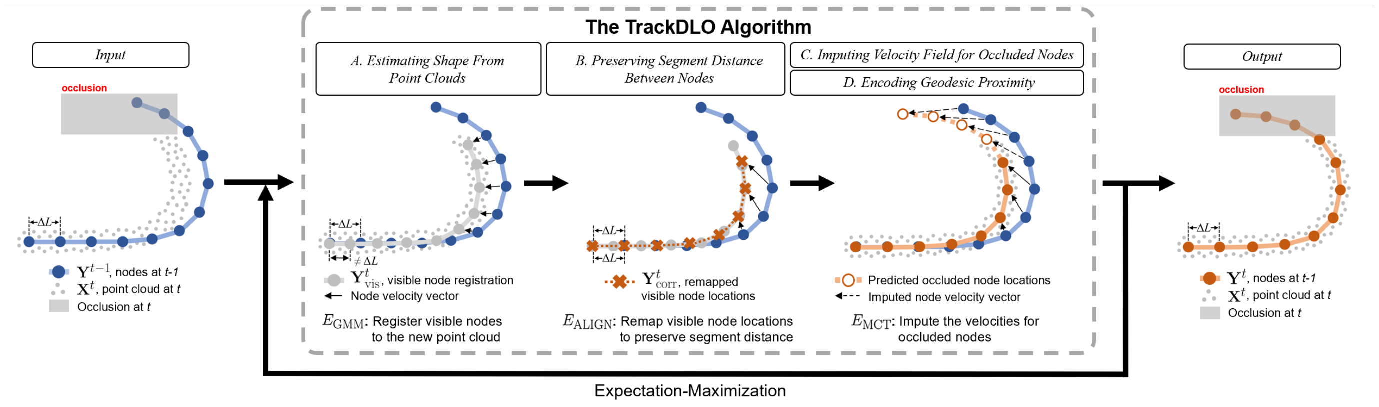

The TrackDLO Algorithm
Input
occlusion
ΔL
$\mathbf{Y}^{t-1}$, nodes at t-1
$\mathbf{X}^t$, point cloud at t
Occlusion at t
A. Estimating Shape From Point Clouds
$\mathbf{Y}^t_{\text{vis}}$, visible node registration
Node velocity vector
E_{GMM}: Register visible nodes to the new point cloud
B. Preserving Segment Distance Between Nodes
$\mathbf{Y}^t_{\text{corr}}$, remapped visible node locations
E_{ALIGN}: Remap visible node locations to preserve segment distance
C. Imputing Velocity Field for Occluded Nodes
D. Encoding Geodesic Proximity
Predicted occluded node locations
Imputed node velocity vector
E_{MCT}: Impute the velocities for occluded nodes
Output
$\mathbf{Y}^t$, nodes at t-1
$\mathbf{X}^t$, point cloud at t
Occlusion at t
Expectation-Maximization

Fig. 2. The TrackDLO algorithm registers visible nodes to a point cloud, preserves segment distance between nodes, imputes the velocity field for occluded nodes, and encodes geometric proximity to accurately track a DLO under occlusion.

engine to update the shape estimate [11]. Structure Preserved Registration (SPR) builds on CPD+Physics by adding modified locally linear embedding for improved accuracy [12]. The Constrained Deformable CPD (CDCPD) algorithm uses GLTP for non-rigid registration, constrains DLO stretching, and detects and recovers from tracking failure [13]. The CDCPD2 algorithm builds on CDCPD by incorporating known correspondences, self-intersection constraints, and obstacle interaction constraints to accurately track under tip and large-scale occlusion [14].

Some occluded DLO tracking methods forego using non-rigid point set registration with CPD, instead using maximum a posteriori estimation for registration and physics simulation for shape updates [10]. More recent work performs DLO tracking under occlusion with particle filtering on a lower-dimensional latent space embedding learned with an autoencoder [15] and learning-based shape estimation under occlusion [20].

These existing methods require additional information from modeling, simulation, visual markers, or contact for accuracy or show inconsistent performance under occlusion.

III. The TrackDLO Algorithm

The TrackDLO algorithm is described in Figure 2. For N *points* in 3D received at time t from a depth sensor, $\mathbf{X}^t_{N\times 3} = (\mathbf{x}^t_1, \ldots, \mathbf{x}^t_N)^T$, TrackDLO represents the DLO shape with a collection of M ordered *nodes*, $\mathbf{Y}^t_{M\times 3} = (\mathbf{y}^t_1, \ldots, \mathbf{y}^t_M)^T$, and the edges connecting adjacent nodes. The measurement $\mathbf{X}^t$ can contain outliers due to noise and it can be incomplete due to occlusion.

The TrackDLO algorithm is initialized by de-projecting a pixel chain output by an occlusion-robust DLO detection algorithm into 3D and sampling nodes equally distributed along the chain [21]. The following subsections describe modifications to the CPD objective function which solves for $\mathbf{Y}^t$ through EM. Gaussian Mixture Model (GMM) clustering with modified membership probability is the basis of non-rigid registration used to estimate the shape of a DLO (Section III-A) [13]. Visible node locations are estimated and remapped to preserve the DLO segment lengths and total length (Section III-B). The velocity field for occluded nodes is imputed using Motion Coherence Theory (MCT) (Section III-C) [16], encoding geodesic proximity to perform accurate tracking in the presence of self-occlusion (Section III-D).

A. Estimating Shape From Point Clouds

The TrackDLO algorithm is based on GMM clustering and computes nodes $\mathbf{Y}^t$ as the centroids of Gaussian distributions from which points $\mathbf{X}^t$ are randomly sampled with isotropic variance σ^2, i.e., $\mathbf{x}^t_n \sim \mathcal{N}(\mathbf{y}^t_m; \sigma^2)$. The mixture model incorporates a uniform distribution with parameter μ to classify outliers in $\mathbf{X}^t$ [17]. Methods derived from CPD assume each Gaussian probability distribution has equal membership probability $p(m) = 1/M$, however CDCPD demonstrates visible nodes are more likely to produce $\mathbf{X}^t$ [13], [17]. For small DLO movement between time steps, TrackDLO estimates the visibility of $\mathbf{y}^t_m$ by finding $\mathbf{x}^{*t}_n$, the closest point to $\mathbf{y}^{t-1}_m$. If $\|\mathbf{x}^{*t}_n - \mathbf{y}^{t-1}_m\| < \tau_{vis}$ for visibility threshold τ_{vis}, $\mathbf{y}^t_m$ is considered visible. Figure 3 visualizes the following steps which check for self-occlusion and prune nodes incorrectly estimated as visible:

1) For each edge in $\mathbf{Y}^{t-1}$, compute the distance between the mid-point of the edge and the camera. Sort edges based on the distance to camera from near to far.
2) Initialize an empty mask with the same dimensions as the RGB image. Project the first edge (nearest to the camera) in the sorted list onto the mask as a line segment with DLO pixel width w. Then, project the next-closest edge onto the mask. Iterating through the sorted edges, if any nodes project onto an existing line segment in the mask, they are occluded by an edge which is closer to the camera.

The TrackDLO algorithm incorporates a constant w, assuming the DLO depth does not vary significantly across different DLO segments. For more complicated 3D shapes, the projected DLO width w should be calculated based on the distance between the current segment and the camera.

The modified membership probability is

$$p(m) = \begin{cases} (1-\mu)p_{\text{vis}}(m) & m < M+1 \\ \mu & m = M+1 \end{cases}. \tag{1}$$

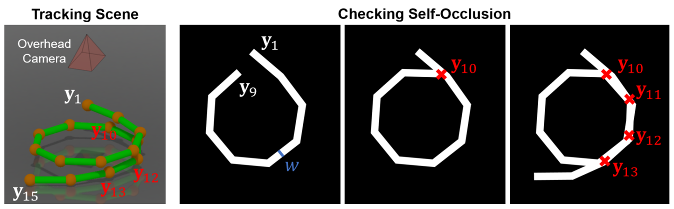


Fig. 3. Each node and edge in $\mathbf{Y}^{t-1}$ is projected onto a mask in order of its distance to the camera. If a node projects onto an existing edge (here, $\mathbf{y}_{10}$ to $\mathbf{y}_{13}$), it is considered occluded by an edge closer to the camera.

Where k_{vis} weights the reliability of the visibility estimation and γ normalizes $p_{\text{vis}}(m)$ so that $\sum_m p_{\text{vis}}(m) = 1$,

$$p_{\text{vis}}(m) = \gamma e^{-k_{\text{vis}} \|\mathbf{x}_n^{*t} - \mathbf{y}_m^{t-1}\|}. \tag{2}$$

Since all visible nodes should share the same membership probability, $\|\mathbf{x}_n^{*t} - \mathbf{y}_m^{t-1}\|$ is set to 0 if $\mathbf{y}_m^{t-1}$ is visible. In the E-step of GMM clustering, the posterior distributions which maximize the likelihood of producing the measured points are computed using Bayes' theorem as

$$p(m|\mathbf{x}_n^t) = \frac{\exp\left(\frac{-\|\mathbf{y}_m^t - \mathbf{x}_n^t\|^2}{2\sigma^2}\right) p_{\text{vis}}(m)}{\sum_{m=1}^{M} \exp\left(\frac{-\|\mathbf{y}_m^t - \mathbf{x}_n^t\|^2}{2\sigma^2}\right) p_{\text{vis}}(m) + \frac{(2\pi\sigma^2)^{3/2}\mu}{(1-\mu)N}}. \tag{3}$$

In the M-step, the new $\mathbf{y}_m^t$ and σ^2 are found by minimizing

$$\begin{aligned} &E_{\text{GMM}}(\mathbf{y}_m^t, \sigma^2) \\ &= \sum_{n=1}^{N} \sum_{m=1}^{M} p(m|\mathbf{x}_n^t) \frac{\|\mathbf{x}_n^t - \mathbf{y}_m^t\|^2}{2\sigma^2} + \frac{3N_p}{2}\log(\sigma^2), \end{aligned} \tag{4}$$

where $N_p = \sum_{n=1}^{N} \sum_{m=1}^{M} p(m|\mathbf{x}_n^t)$.

Node registration with GMM clustering does not provide correspondence information between two consecutive frames. Furthermore, if $\mathbf{X}^t$ is incomplete due to occlusion, $\mathbf{Y}^t$ will only be predicted in regions where the DLO is visible, leading to incorrect shape estimates. These challenges were addressed by the CDCPD algorithm, enabling accurate DLO tracking under minor occlusion. However, when one tip of the DLO is occluded and $\mathbf{X}^t$ is incomplete, current algorithms compute compressed shapes with lengths shorter than the total length. The CDCPD2 algorithm solves the tip occlusion problem using diminishing rigidity to describe the positions of nodes at a tip grasped by a robot gripper [14]. However, robot motion information may be unavailable for tracking. TrackDLO addresses the tip occlusion problem without gripper motion information using length preservation and MCT as discussed in subsequent sections.

B. *Preserving Segment Distance Between Nodes*

Many DLOs admit a limited amount of extension or compression and the segment length between adjacent nodes, ΔL, stays constant. At each time step, the visible nodes $\mathbf{Y}_{\text{vis}}^{t-1}$ are first registered to $\mathbf{X}^t$ using GLTP [19], producing $\mathbf{Y}_{\text{vis}}^t$. Connecting adjacent nodes in $\mathbf{Y}_{\text{vis}}^t$ creates a piecewise linear (PWL) curve shape for the visible portion of the DLO and estimates the positions of the tips, $\mathbf{y}_1^t$ and $\mathbf{y}_M^t$. The estimate

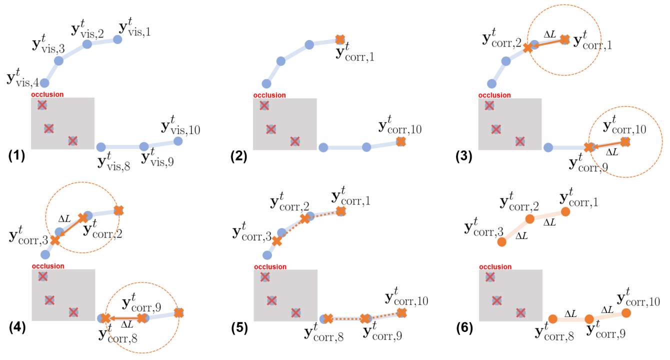


Fig. 4. Length preservation remaps visible nodes onto the piecewise linear curve connecting the GLTP-registered visible nodes, $\mathbf{Y}_{\text{vis}}$. In this illustration, $\mathbf{y}_{\text{vis},1}^t$ through $\mathbf{y}_{\text{vis},4}^t$ and $\mathbf{y}_{\text{vis},7}^t$ through $\mathbf{y}_{\text{vis},9}^t$ are the GLTP registration output. Starting at the visible tip nodes $\mathbf{y}_{\text{vis},1}^t$ and $\mathbf{y}_{\text{vis},9}^t$, six nodes are remapped onto the curve. The set of correspondence pairs computed from this frame are $\mathcal{C} = \{(\mathbf{y}_{\text{corr},1}^t, \mathbf{y}_1^{t-1}), (\mathbf{y}_{\text{corr},2}^t, \mathbf{y}_2^{t-1}), (\mathbf{y}_{\text{corr},3}^t, \mathbf{y}_3^{t-1}), (\mathbf{y}_{\text{corr},8}^t, \mathbf{y}_8^{t-1}), (\mathbf{y}_{\text{corr},9}^t, \mathbf{y}_9^{t-1}), (\mathbf{y}_{\text{corr},10}^t, \mathbf{y}_{10}^{t-1})\}$.

of the visible nodes with segment distance preserved, $\mathbf{Y}_{\text{corr}}^t$, is computed by remapping $\mathbf{Y}_{\text{vis}}^t$ onto the PWL curve at distances ΔL from each other as shown in Figure 4. This creates a set of correspondence pairs $\mathcal{C} = \{(\mathbf{y}_{\text{corr},m}^t, \mathbf{y}_m^{t-1})\}$. Node traversal starts from the visible tip nodes and moves toward opposing ends, fitting as many nodes as possible until reaching an occluded region. If neither of the tip nodes is visible, traversal starts from the node which moved the least between $\mathbf{Y}_{\text{vis}}^{t-1}$ and $\mathbf{Y}_{\text{vis}}^t$. Since GLTP is accurate for minor occlusion, occluded nodes are included in registration and traversal if they are between two visible nodes less than a geodesic distance d_{vis} apart. To align $\mathbf{Y}^t$ to $\mathbf{Y}_{\text{corr}}^t$, an $E_{\text{ALIGN}}(\mathbf{y}_m^t, \sigma^2)$ term is added to (4) as

$$E_{\text{ALIGN}}(\mathbf{y}_m^t, \sigma^2) = \frac{\alpha}{2} \sum_{m=1}^{M} J(m) \|\mathbf{y}_{\text{corr},m}^t - \mathbf{y}_m^t\|^2, \tag{5}$$

where the parameter α weights the amount of alignment and $J(m) = 1$ if and only if $(\mathbf{y}_{\text{corr},m}^t, \mathbf{y}_m^{t-1}) \in \mathcal{C}$.

C. *Imputing Velocity Field for Occluded Nodes*

Occluded nodes will contribute a small E_{GMM} cost due to a small p_{vis} and they will not contribute any E_{ALIGN} cost for position regularization. To address occlusion, movement of the occluded node is imputed from the movement of visible nodes using Motion Coherence Theory (MCT) which regularizes the spatial velocity field [16].

1) Deriving a Cost for MCT: Given node positions $\mathbf{Y}^{t-1}$ and $\mathbf{Y}^t$ from consecutive time steps, the MCT defines a spatial velocity field $v(\mathbf{Y}^{t-1}) = \mathbf{Y}^t - \mathbf{Y}^{t-1}$. Nodes close to each other move coherently through the smoothest possible spatial velocity field. For a variable in the spatial domain, $\mathbf{z}$, the smoothness of the velocity field, $v(\mathbf{z})$, is

$$E_{\text{MCT}}(v(\mathbf{z})) = \frac{\lambda}{2} \int_{\mathbb{R}^D} \sum_{l=0}^{\infty} c_l \|\mathbf{D}^l v(\mathbf{z})\|^2 d\mathbf{z}, \tag{6}$$

where c_l weights each lth order of the derivative operator $\mathbf{D}$ and $\mathbf{D}^{2l+1}v = \nabla(\nabla^{2l}v)$. Adding E_{MCT} to (4) and noting $\mathbf{y}_m^t = \mathbf{y}_m^{t-1} + v(\mathbf{y}_m^{t-1})$, the total cost becomes

$$\begin{aligned} E(v(\mathbf{z}),\sigma^2) &= E_{\text{GMM}} + E_{\text{ALIGN}} + E_{\text{MCT}} \\ &= \sum_{n=1}^{N}\sum_{m=1}^{M}\frac{1}{2\sigma^2}p(m|\mathbf{x}_n^t)\|\mathbf{x}_n^t - \mathbf{y}_m^t\|^2 + \frac{3N_p}{2}\log(\sigma^2) \\ &+ \frac{\alpha}{2}\sum_{m=1}^{M} J(m)\|\mathbf{y}_m^t - \mathbf{y}_{\text{corr},m}^t\|^2 \\ &+ \frac{\lambda}{2}\int_{\mathbb{R}^D}\sum_{l=0}^{\infty} c_l\|\mathbf{D}^l v(\mathbf{z}))\|^2 d\mathbf{z} \end{aligned} . \quad (7)$$

2) Deriving a Kernel for MCT Regularization: The $v(\mathbf{z})$ that minimizes (7) takes the form

$$v(\mathbf{z}) = \sum_{m=1}^{M}\mathbf{w}_m^t G(\|\mathbf{z} - \mathbf{y}_m^{t-1}\|), \quad (8)$$

where $\mathbf{w}_m^t$ can be found through EM and $G(\|\mathbf{z}\|)$ is the Green's function of operator $\mathbf{L} = \sum_{l=0}^{\infty}(-1)^l c_l \nabla^{2l}$ [16], [22]. Applying the definition of Green's function, $\mathbf{L}G(\|\mathbf{z}\|) = \delta(\mathbf{z})$, the Fourier Transform of $G(\|\mathbf{z}\|)$ must satisfy

$$\tilde{G}(\|\mathbf{s}\|) = \mathcal{F}\{G(\|\mathbf{z}\|)\} = \frac{1}{\sum_{l=0}^{\infty} c_l\|\mathbf{s}\|^{2l}}. \quad (9)$$

Any $G(\|\mathbf{z}\|)$ with a $\tilde{G}(\|\mathbf{s}\|)$ satisfying (9) can be used in EM to solve for the $v(\mathbf{z})$ that minimizes (7). The MCT proposes a Gaussian kernel, where $G(\|\mathbf{z}\|) = e^{-\|\mathbf{z}\|^2/(2\beta^2)}/(\sqrt{2\pi}\beta)$ and $\tilde{G}(\|\mathbf{s}\|) = e^{-\beta^2\|\mathbf{s}\|^2/2}$. Through Taylor expansion,

$$\tilde{G}(\|\mathbf{s}\|) = \frac{1}{\sum_{l=0}^{\infty}\frac{\beta^{2l}}{2^l l!}(\|\mathbf{s}\|)^{2l}} \quad (10)$$

Therefore, for a Gaussian kernel, $c_l = \frac{\beta^{2l}}{2^l l!}$ with parameter β controlling the strength of MCT regularization. For a small β, MCT regularization will be weak and estimated node positions will be jittery. For a large β, MCT regularization will be strong and the relative position between nodes will not change.

Since c_l will always be non-zero for a Gaussian kernel, it penalizes all spatial derivatives of the velocity field. This produces a velocity field where all higher order derivatives are as smooth as possible. In practice, smoothing all higher order derivatives leads to error accumulation for tracking under occlusion. This work proposes a new kernel with

$$G(\|\mathbf{z}\|) = \frac{1}{2\beta^2}e^{-2\|\mathbf{z}\|/\beta}(2\|\mathbf{z}\| + \beta) \quad (11)$$

$$\tilde{G}(\|\mathbf{s}\|) = \frac{1}{1 + (\beta^2/2)\|\mathbf{s}\|^2 + (\beta^4/16)\|\mathbf{s}\|^4}. \quad (12)$$

This kernel, referred to as a "2nd Order kernel," has $c_0 = 1$, $c_1 = \beta^2/2$, $c_2 = \beta^4/16$, and $c_{l>2} = 0$. The 2nd Order kernel only penalizes the first and second derivatives of the velocity field which reduces noise amplification for tracking under occlusion as compared to the Gaussian kernel. These kernels are compared in Section IV-C.

3) Deriving the Solution for the Total Cost: This section derives an analytic closed-form solution for $\mathbf{w}_m^t$ and σ^2 in the EM algorithm using MCT regularization and the following notations:

- $\mathbf{W}_{M\times 3}$–the collection of weights, $(\mathbf{w}_1^t, \ldots, \mathbf{w}_M^t)^T$,
- $\mathbf{P}_{M\times N}$–the posteriori probability matrix with $\mathbf{P}(m,n) = p(m|\mathbf{x}_n^t)$,
- $\mathbf{G}_{M\times M}$–the kernel matrix with $\mathbf{G}(i,m) = G(d_{\mathbf{y}_i^t,\mathbf{y}_m^t})$, where d is a measure of distance (discussed in Section III-D) and G is a valid kernel,
- $\mathbf{J}_{M\times M}$–an identity matrix with an m^{th} row of zeros if $(\mathbf{y}_{\text{corr},m}^t, \mathbf{y}_m^t) \notin \mathcal{C}$,
- $\mathbf{Y}_{\text{corr}}$–a $M \times 3$ zero matrix with its m^{th} row equals to $\mathbf{y}_{\text{corr},m}^t$ if $(\mathbf{y}_{\text{corr},m}^t, \mathbf{y}_m^t) \in \mathcal{C}$,
- $\text{d}(\mathbf{a})$–the diagonal matrix constructed from vector $\mathbf{a}$,
- $\text{tr}(\mathbf{m})$–the trace of matrix $\mathbf{m}$, and
- $\mathbf{1}$–a column vector of ones.

For readability, denote $\mathbf{X}^t$ as $\mathbf{X}$, $\mathbf{Y}^t$ as $\mathbf{Y}$, $\mathbf{Y}^{t-1}$ as $\mathbf{Y}_0$, and $\mathbf{W}^t$ as $\mathbf{W}$. In the frequency domain, E_{MCT} is [23]

$$E_{\text{MCT}} = \int_{\mathbb{R}^D}|\tilde{v}(\mathbf{s})|^2/\tilde{G}(\|\mathbf{s}\|)d\mathbf{s}, \quad (13)$$

which is equivalent to $\text{tr}(\mathbf{W}^T\mathbf{G}\mathbf{W})$ in matrix form. The solutions $\mathbf{W}$ and σ^2 are computed from taking the partial derivatives $\frac{\partial E}{\partial \mathbf{W}}$ and $\frac{\partial E}{\partial \sigma^2}$ and setting them to zero:

$$\begin{aligned} \mathbf{W} = &\left(\text{d}(\mathbf{P1})\mathbf{G} + \lambda\sigma^2\mathbf{I} + \alpha\mathbf{J}\mathbf{G}\right)^{-1} \\ &\cdot(\mathbf{P}\mathbf{X} - \text{d}(\mathbf{P1})\mathbf{Y}_0 + \alpha\mathbf{J}(\mathbf{Y}_{\text{corr}} - \mathbf{Y}_0)) \end{aligned} \quad (14)$$

$$\begin{aligned} \sigma^2 = &\frac{1}{\mathbf{1}^T\mathbf{P1}D}(\text{tr}(\mathbf{X}^T\text{d}(\mathbf{P}^T\mathbf{1})\mathbf{X}) - 2\text{tr}((\mathbf{P}\mathbf{X})^T\mathbf{Y}) \\ &+ \text{tr}(\mathbf{Y}^T\text{d}(\mathbf{P1})\mathbf{Y})) \end{aligned} . \quad (15)$$

The E-step and M-step are performed alternately until convergence of the EM algorithm to solve for $\mathbf{Y} = \mathbf{Y}_0 + \mathbf{G}\mathbf{W}$.

D. Encoding Geodesic Proximity

The MCT requires the velocity field to be smooth, where nodes spatially close to each other move in similar directions with similar speeds. In the node velocity

$$\begin{aligned} v(\mathbf{y}_m^{t-1}) &= \sum_{i=1}^{M}\mathbf{G}(m,i)\mathbf{W}^t(i,\cdot) \\ &= \sum_{i=1}^{M} G(d_{\mathbf{y}_i^{t-1},\mathbf{y}_m^{t-1}})\mathbf{W}^t(i,\cdot), \end{aligned} \quad (16)$$

$d_{\mathbf{y}_i^{t-1},\mathbf{y}_m^{t-1}}$ is the distance between nodes $\mathbf{y}_i^{t-1}$ and $\mathbf{y}_m^{t-1}$ given some measure of distance. Since $G(\|\mathbf{z}\|)$ decreases as $\|\mathbf{z}\|$ increases, $G(d_{\mathbf{y}_i^{t-1},\mathbf{y}_m^{t-1}})$ is small if $\mathbf{y}_i^{t-1}$ is far from $\mathbf{y}_m^{t-1}$. This produces a small $G(d_{\mathbf{y}_i^{t-1},\mathbf{y}_m^{t-1}})\mathbf{W}^t(i,\cdot)$, indicating the movement of $\mathbf{y}_i^{t-1}$ has little influence on the movement of $\mathbf{y}_m^{t-1}$.

The metric Euclidean distance $d^E_{\mathbf{y}_i^t,\mathbf{y}_m^t} = \|\mathbf{y}_m^t - \mathbf{y}_i^t\|$ is a common choice for $d_{\mathbf{y}_i^t,\mathbf{y}_m^t}$, however topological distances more accurately represent proximity across surfaces. For a self-occluding DLO, if the top part moves, the bottom part should move a small amount if at all. All nodes near the crossing have small Euclidean proximities from each other even though they are from distant parts of the DLO topology; the kernel $\mathbf{G}(d^E_{\mathbf{y}_i^t,\mathbf{y}_m^t})$ causes strong interaction among these nodes close in Euclidean space. The TrackDLO algorithm uses the topological geodesic distance, $d^G_{\mathbf{y}_i^t,\mathbf{y}_m^t}$, in place of $d^E_{\mathbf{y}_i^t,\mathbf{y}_m^t}$.

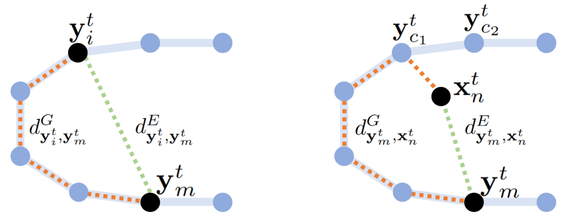

$\mathbf{y}_i^t$
$d^G_{\mathbf{y}_i^t,\mathbf{y}_m^t}$
$d^E_{\mathbf{y}_i^t,\mathbf{y}_m^t}$
$\mathbf{y}_m^t$
$\mathbf{y}_{c_1}^t$
$\mathbf{y}_{c_2}^t$
$\mathbf{x}_n^t$
$d^G_{\mathbf{y}_m^t,\mathbf{x}_n^t}$
$d^E_{\mathbf{y}_m^t,\mathbf{x}_n^t}$
$\mathbf{y}_m^t$

Fig. 5. The geodesic distance $d^G_{\mathbf{y}_i^t,\mathbf{y}_m^t}$ (orange) better represents the distance between nodes describing the shape of a DLO than the Euclidean distance $d^E_{\mathbf{y}_i^t,\mathbf{y}_m^t}$ (green). (Left) Node-to-node $d^G_{\mathbf{y}_i^t,\mathbf{y}_m^t}$ and $d^E_{\mathbf{y}_i^t,\mathbf{y}_m^t}$. (Right) Node-to-point $d^G_{\mathbf{y}_m^t,\mathbf{x}_n^t}$ and $d^E_{\mathbf{y}_m^t,\mathbf{x}_n^t}$.

Using geodesic distance, a node on the top part of the DLO and a node on the bottom part of the DLO near the crossing do not significantly influence the motion of each other because they are far in geodesic proximity. The node-to-node geodesic distance, shown in Figure 5, is

$$d^G_{\mathbf{y}_i^t,\mathbf{y}_m^t} = \begin{cases} \sum_{j=m}^{i-1} \|\mathbf{y}_{j+1}^t - \mathbf{y}_j^t\| & \text{if } m \le i \\ \sum_{j=i}^{m-1} \|\mathbf{y}_{j+1}^t - \mathbf{y}_j^t\| & \text{if } m > i \end{cases}. \tag{17}$$

Similarly, two nodes close in Euclidean distance to a crossing but from different parts of the DLO should have different proximities to points near the crossing. Where $\mathbf{y}_{c_1}^t$ and $\mathbf{y}_{c_2}^t$ are the two nodes with the shortest Euclidean distances to point $\mathbf{x}_n^t$, the node-to-point geodesic distance $d^G_{\mathbf{y}_m^t,\mathbf{x}_n^t}$ is

$$d^G_{\mathbf{y}_m^t,\mathbf{x}_n^t} = \begin{cases} d^E_{\mathbf{y}_{c_1}^t,\mathbf{x}_n^t} + d^G_{\mathbf{y}_{c_1}^t,\mathbf{y}_m^t} & d^G_{\mathbf{y}_m^t,\mathbf{y}_{c_1}^t} \le d^G_{\mathbf{y}_m^t,\mathbf{y}_{c_2}^t} \\ d^E_{\mathbf{y}_{c_2}^t,\mathbf{x}_n^t} + d^G_{\mathbf{y}_{c_1}^t,\mathbf{y}_m^t} & d^G_{\mathbf{y}_m^t,\mathbf{y}_{c_1}^t} > d^G_{\mathbf{y}_m^t,\mathbf{y}_{c_2}^t} \\ d^E_{\mathbf{y}_m^t,\mathbf{x}_n^t} & m \in \{c_1, c_2\} \end{cases}. \tag{18}$$

The node-to-point geodesic distance $d^G_{\mathbf{y}_m^t,\mathbf{x}_n^t}$ replaces the Euclidean distance $\|\mathbf{y}_m^t - \mathbf{x}_n^t\|$ in (3).

IV. Experiments

In experiments analyzing the accuracy of TrackDLO, CDCPD2 with and without optional gripper information, CDCPD, and GLTP, TrackDLO achieves the lowest node-to-PWL curve frame error (Section IV-A). The performance impacts of segment distance preservation (Section IV-B), choice of kernel (Section IV-C), and distance metric (Section IV-D) for TrackDLO are reported along with a comparison of computation time (Section IV-E). All experiments used an Intel RealSense d435 camera with $k_{\text{vis}} = 500, \beta = 0.5, \lambda = 5{\times}10^4$, $\alpha = 3$, $w = 20$ pixels, $\tau_{\text{vis}} = 5\text{mm}$, $d_{\text{vis}} = 4\text{cm}$, and convergence of EM when the average node position change between iterations is less than 0.2mm.

A. Tracking Error

Experiments comparing the performance of TrackDLO, CDCPD2 with and without optional gripper information, CDCPD, and GLTP under occlusion used the frame error metric shown in Figure 6 and defined as follows. The set of points $\text{PWL}(\mathbf{Y})$ in the PWL curve representation of an ordered node sequence $\mathbf{Y}$ includes both the nodes themselves and all points in the line segments connecting consecutive nodes. The point-set distance is

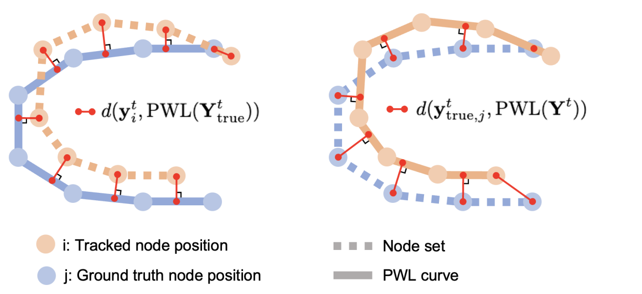

$d(\mathbf{y}_i^t, \text{PWL}(\mathbf{Y}_{\text{true}}^t))$
$d(\mathbf{y}_{\text{true},j}^t, \text{PWL}(\mathbf{Y}^t))$
● i: Tracked node position
● j: Ground truth node position
Node set
PWL curve

Fig. 6. The frame error is the average of the errors from the left and right images. (Left) Node-to-PWL curve $\varepsilon(\mathbf{Y}^t, \mathbf{Y}_{\text{true}}^t)$. (Right) Node-to-PWL curve $\varepsilon(\mathbf{Y}_{\text{true}}^t, \mathbf{Y}^t)$.

$$d(\mathbf{y}, \mathcal{S}) = \inf_{\mathbf{y}' \in \mathcal{S}} \|\mathbf{y} - \mathbf{y}'\| \tag{19}$$

where $\mathbf{y}$ is a point and $\mathcal{S}$ is a set of points. The per-node error between two node sequences is

$$\varepsilon(\mathbf{Y}^t, \mathbf{Y}_{\text{true}}^t) = \frac{1}{M} \sum_{\mathbf{y}_i^t \in \mathbf{Y}^t} d(\mathbf{y}_i^t, \text{PWL}(\mathbf{Y}_{\text{true}}^t)), \tag{20}$$

and the frame error metric, mirroring the per-node error to guarantee symmetry, is

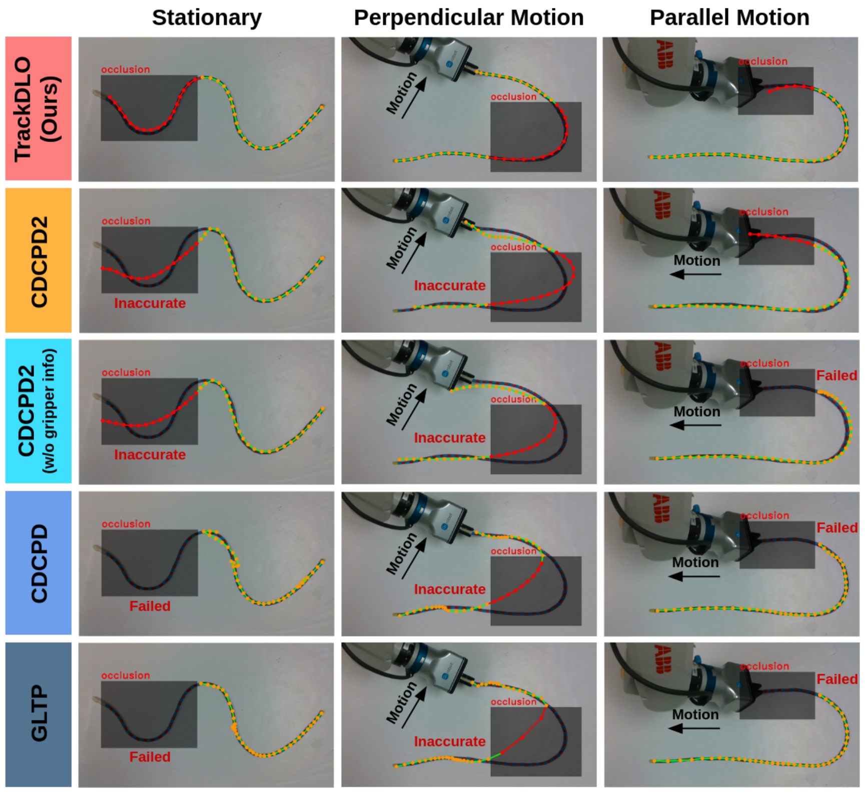
$$\mathcal{E}(\mathbf{Y}_{\text{true}}^t, \mathbf{Y}^t) = \frac{1}{2} \left(\varepsilon(\mathbf{Y}_{\text{true}}^t, \mathbf{Y}^t) + \varepsilon(\mathbf{Y}^t, \mathbf{Y}_{\text{true}}^t) \right). \tag{21}$$

Evaluation was performed for three scenarios:

1) *Stationary*–This scenario tests tracking error accumulation and length preservation under tip occlusion.
2) *Perpendicular Motion*–This scenario tests DLO tracking accuracy when both tips are visible and the mid-section is occluded.
3) *Parallel Motion*–This scenario tests DLO length preservation and tracking accuracy when one tip moves through occlusion.

For each scenario, RGB-D image and point data were saved in a Robot Operating System (ROS) bag file. Occlusion was injected by removing pixels within a bounding box area in the DLO segmentation mask. The blue rope DLO was evenly marked with red tape which was segmented by thresholding on the red and blue colors. In each of the red and blue segmentation masks, contour filtering and blob detection identified the blue and red segments along the DLO and keypoint detection returned their centroids. These centroids were combined to form the ground truth nodes, $\mathbf{Y}_{\text{true}}^t$, which were compared to the tracked nodes in $\mathbf{Y}^t$. Note the markers on the DLO were only used for evaluation and were not required by any algorithm for tracking. Each experiment was repeated 10 times for each algorithm in each scenario.

Evaluation results are reported in Figure 7. In the *Stationary* scenario, the occlusion size is scaled as a percentage of the number of nodes along the DLO and the DLO is occluded for 25 seconds after initialization. The TrackDLO algorithm achieves the lowest average final frame error as a function of percentage of occlusion for the *Stationary* scenario. In the *Perpendicular Motion* scenario, occlusion is injected in the



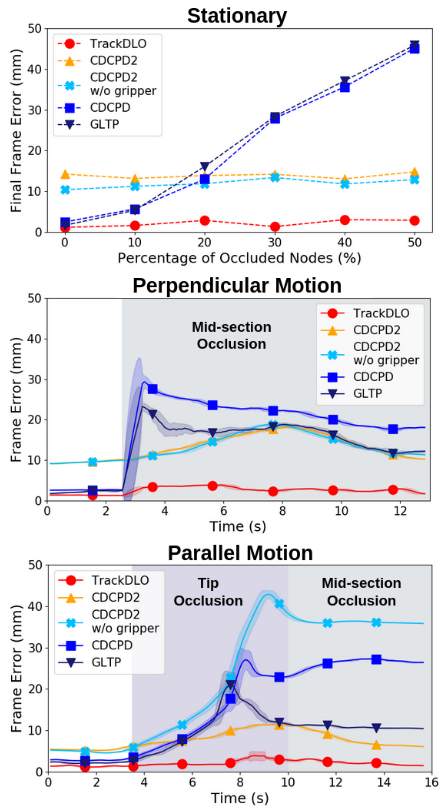


Fig. 7. Compared to CDCPD2 with and without gripper information, CDCPD, and GLTP, TrackDLO accurately estimates the state of the DLO under scaled, tip, and mid-section occlusion. Among these algorithms, TrackDLO had the lowest frame error.

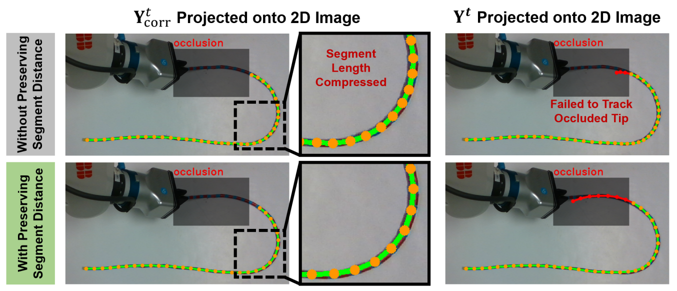

$\mathbf{Y}^t_{\text{corr}}$ Projected onto 2D Image
$\mathbf{Y}^t$ Projected onto 2D Image
Without Preserving Segment Distance
With Preserving Segment Distance
occlusion
Segment Length Compressed
Failed to Track Occluded Tip

Fig. 8. Segment distance preservation prevents compression of the length of the DLO, a failure scenario which is most prominent under tip occlusion.

mid-section of the DLO three seconds after tracking begins before the object moves and remains there until the end of the bag file. For the CDCPD2, CDCPD, and GLTP algorithms, the error increases after the injection of occlusion and decreases as the tracking estimates begin to catch up with the DLO state. The TrackDLO algorithm achieves the lowest average frame error in this scenario. In the *Parallel Motion* scenario, the DLO is initially fully visible and the gripper moves the DLO tip through an occluded region. Tracking error increases for all algorithms until the DLO tip becomes visible again during the mid-section occlusion period. The TrackDLO algorithm achieves the lowest frame error in this scenario as well.

B. Segment Distance Preservation

Preserving segment distance between nodes is important for DLO tracking under occlusion. To demonstrate this, tracking is performed on the *Parallel Motion* scenario with and without segment distance preservation. Without preserving segment distance, $\mathbf{Y}^t_{\text{corr}}$ is simply $\mathbf{Y}^t_{\text{vis}}$ and nodes are not traversed to remap their positions. This produces compressed segment lengths between nodes and a failure to accurately estimate the location of the tip under occlusion as shown in Figure 8. Remapping the node positions in $\mathbf{Y}^t_{\text{vis}}$ prevents producing compressed tracking results under occlusion.

C. Kernel Selection

Analysis of the following kernels shows the influence of kernel on velocity imputation:

1) Laplacian–$G_1(\mathbf{z}) = \frac{1}{\sqrt{2}\beta} e^{-\sqrt{2}\|\mathbf{z}\|/\beta}$
2) 2nd Order–$G_2(\mathbf{z}) = \frac{1}{2\beta^2} e^{-2\|\mathbf{z}\|/\beta}(2\|\mathbf{z}\| + \beta)$
3) 3rd Order–$G_3(\mathbf{z}) = \frac{3}{16\beta^3} e^{-\sqrt{6}\|\mathbf{z}\|/\beta}(2\sqrt{6}\|\mathbf{z}\|^2 + 6\beta\|\mathbf{z}\| + \sqrt{6}\beta^2)$
4) Gaussian–$G_4(\mathbf{z}) = \frac{1}{\sqrt{2\pi}\beta} e^{-\|\mathbf{z}\|^2/(2\beta^2)}$

These kernels have the below c_l values for E_{MCT}:

1) $c_0 = 1,\ c_1 = \frac{\beta^2}{2}$, and $c_{l>1} = 0$
2) $c_0 = 1,\ c_1 = \frac{\beta^2}{2},\ c_2 = \frac{\beta^4}{16}$, and $c_{l>2} = 0$
3) $c_0 = 1,\ c_1 = \frac{\beta^2}{2},\ c_2 = \frac{\beta^4}{12},\ c_3 = \frac{\beta^6}{216}$, and $c_{l>3} = 0$
4) $c_l = \frac{\beta^{2l}}{2^l l!}$

The Laplacian kernel minimizes the velocity field $\|\mathbf{D}^0 v\|^2$ and its first derivative $\|\mathbf{D}^1 v\|^2$. The 2nd and 3rd Order kernels include up to $\|\mathbf{D}^2 v\|^2$ and $\|\mathbf{D}^3 v\|^2$, respectively. The Gaussian kernel includes all derivatives of the spatial velocity for

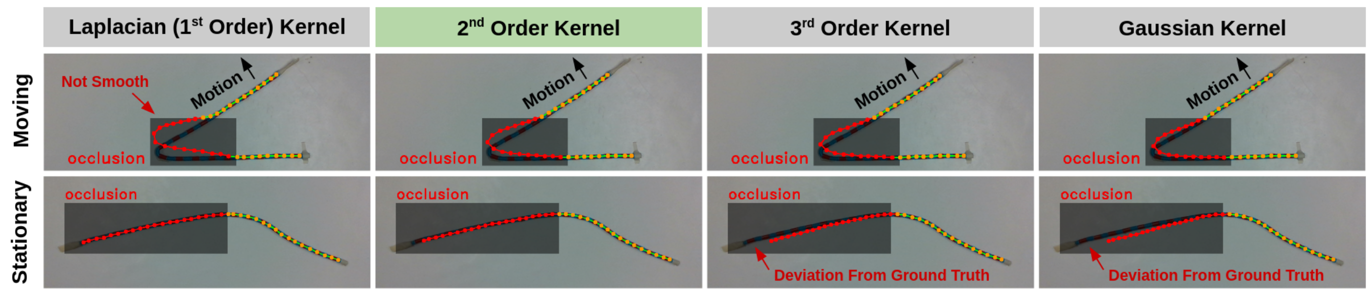


Fig. 9. The Laplacian kernel does not guarantee a smooth enough velocity field when tracking a moving DLO, and the Gaussian kernel amplifies noise and deviations between the estimated and true visible node positions for tracking a stationary DLO. The 2nd Order kernel balances smoothness with noise amplification in tracking a moving and a stationary DLO under occlusion.

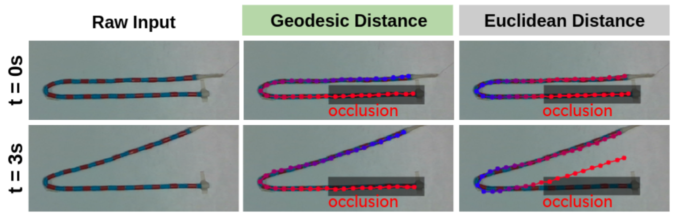


Fig. 10. The color of visible nodes is scaled based on proximity to the nearest occluded node from red (close) to blue (far) using geodesic and Euclidean distances. Using geodesic distance, the movement of the top part of the object has little effect on the movement of the bottom (occluded) part. Using Euclidean distance, there is a large interaction between moving nodes on the top part and stationary nodes on the bottom part.

TABLE I
FRAME ERROR OF DIFFERENT KERNELS (IN MM)

Scenario	Laplacian	**2nd Order (Ours)**	3rd Order	Gaussian
Moving	6.13	4.30	3.56	**2.93**
Stationary	**1.27**	1.83	4.70	6.33
Average	3.70	**2.78**	4.13	4.63

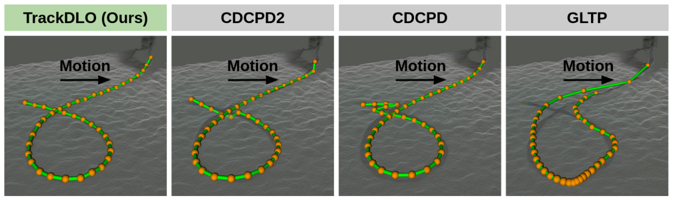


Fig. 11. Using both node-to-node geodesic proximity from (17) and node-to-point geodesic distance from (18), TrackDLO demonstrates comparable performance to CDCPD2 for tracking a self-occluding DLO with the correct topology. The CDCPD and GLTP algorithms use Euclidean distance and fail to resolve the occlusion.

TABLE II
TRACKING TIMING COMPARISON (IN MS)

Algorithm	**TrackDLO (Ours)**	CDCPD2	CDCPD	GLTP
Tracking Step	9.20	11.42	18.14	**5.43**

smoothing. Comparison of kernels for imputing node velocity for occluded nodes is performed in two scenarios:

1) *Moving*–This scenario tests the accuracy of velocity imputation for a DLO moving under occlusion.
2) *Stationary*–This scenario tests the accumulated deviation between $\mathbf{Y}^t$ and $\mathbf{Y}^t_{\text{true}}$ for a stationary DLO.

The tracking errors in Table I for the *Moving* and *Stationary* scenarios are the averages over 10 runs on these ROS bag data. The average error for each kernel is the average of the *Moving* and *Stationary* error. As including more derivatives imposes more smoothness on the velocity field, the Gaussian kernel produces the most accurate results for the *Moving* scenario. Here, the Laplacian kernel still enables finding correspondence between nodes in consecutive frames, but it does not impose enough smoothness to obtain accurate velocity imputation for occluded nodes. In the *Stationary* scenario, there is no limit on the imputed velocity for an occluded DLO tip. When higher-order derivatives are included in the kernel, deviations in the positions of the visible $\mathbf{Y}^t$ from $\mathbf{Y}^t_{\text{true}}$ heavily influence the position estimates of the occluded $\mathbf{Y}^t$ even if the object does not move. After 5 seconds of occlusion, the occluded portion of $\mathbf{Y}^t$ estimated with the 3rd Order and Gaussian kernels deviates from $\mathbf{Y}^t_{\text{true}}$.

This experiment highlights the trade-off between smoothness of velocity imputation and robustness of velocity imputation to deviations of visible $\mathbf{Y}^t$ from $\mathbf{Y}^t_{\text{true}}$. The 2nd Order kernel balances this trade-off to impute a reasonable velocity field and the lowest average frame error across the *Moving* and *Stationary* scenarios.

D. Geodesic and Euclidean Distance

Tracking DLOs with geodesic distance outperforms tracking with Euclidean distance. Since nodes close in proximity move with similar directions and speeds in the MCT, the definition of proximity affects velocity imputation. To demonstrate this, one half of a DLO initially folded is slowly moved upwards while the other half does not move as shown in Figure 10. When the stationary part is occluded, TrackDLO with geodesic distance accurately estimates the velocity of the occluded stationary part as being small, however TrackDLO with Euclidean distance estimates its velocity as being similar to the velocity of the moving part.

Combining node-to-point geodesic distance with node-to-node geodesic distance resolves crossings during self-occlusion. TrackDLO demonstrates comparable performance to CDCPD2 which uses node-to-node geodesic distance and a self-intersection constraint to address self-occlusion. The CDCPD and GLTP algorithms use Euclidean distance and fail to resolve crossings as shown in Figure 11.

E. Tracking Timing

Timing experiments were conducted to compare the speeds of the TrackDLO, CDCPD2, CDCPD, and GLTP algorithms. The times reported in Table II were obtained on a computer with a Ryzen Threadripper 3960X CPU and 64 GB RAM. The CDCPD algorithm is implemented in Python while TrackDLO, CDCPD2, and GLTP are implemented in C++ using original repositories where available. Times include computation time for each algorithm after receiving data. Times exclude interfacing with ROS, segmenting object masks, and downsampling point clouds. Each algorithm was run on the *Perpendicular Motion* and *Parallel Motion* ROS bag files for 10 trials. Times are the average computation times across each algorithm run on each frame of the 10 trials of the two motion scenarios. The TrackDLO algorithm achieves the second-fastest average computation time.

V. Conclusions and Future Work

This work introduced TrackDLO, a real-time, vision-only algorithm for occluded DLO tracking. TrackDLO is robust under three types of occlusion and produced the lowest tracking error when compared to the CDCPD2, CDCPD, and GLTP algorithms. Source code, data, and a video of examples of DLO tracking are released for benchmarking. The limitations of TrackDLO include the requirements of good depth resolution, small motion between frames, and motion of occluded nodes to be reflected in the motion of visible nodes. Future work could track multiple DLOs as they move in cluttered environments [24]–[27], track a DLO as multiple sections of it are occluded (where the occlusion size is significantly greater than d_{vis}), or integrate DLO tracking into closed-loop DLO shape control [4], [28].

Acknowledgements

The authors thank Dylan Colli for assistance with the CDCPD2 repository, Paulo V.K. Borges for providing feedback on this manuscript, and the members of the Representing and Manipulating Deformable Linear Objects project for their support. Jingyi Xiang performed part of this work with the Illinois Space Grant Consortium Undergraduate Research Opportunity Program, and NASA Space Technology Graduate Research Opportunity award 80NSSC21K1292 supported Holly Dinkel.

References

[1] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-Supervised Learning of State Estimation for Manipulating Deformable Linear Objects," in *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, Apr. 2020, pp. 2372–2379.

[2] M. Yan, G. Li, Y. Zhu, and J. Bohg, "Learning Topological Motion Primitives for Knot Planning," in *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2020.

[3] H. Yin, A. Varava, and D. Kragic, "Modeling, Learning, Perception, and Control Methods for Deformable Object Manipulation," in *Sci. Rob.*, vol. 6, May 2021, pp. 1–16.

[4] M. Yu, H. Zhong, and X. Li, "Shape Control of Deformable Linear Objects with Offline and Online Learning of Local Linear Deformation Models," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 1337–1343.

[5] S. Jin, W. Lian, C. Wang, M. Tomizuka, and S. Schaal, "Robotic Cable Routing with Spatial Representation," in *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, Apr. 2022, pp. 5687–5694.

[6] B. Lu, H. K. Chu, and L. Cheng, "Dynamic Trajectory Planning for Robotic Knot Tying," in *IEEE Int. Conf. Real-Time Comput. Robot. (RCAR)*, 2016, pp. 180–185.

[7] V. Viswanath, J. Grannen, P. Sundaresan, B. Thananjeyan, A. Balakrishna, E. Novoseller, J. Ichnowski, M. Laskey, J. E. Gonzalez, and K. Goldberg, "Disentangling Dense Multi-Cable Knots," in *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2021, pp. 3731–3738.

[8] J. Guo, J. Zhang, D. Wu, Y. Gai, and K. Chen, "An Algorithm Based on Bidirectional Searching and Geometric Constrained Sampling for Automatic Manipulation Planning in Aircraft Cable Assembly," *J. Manuf. Syst.*, vol. 57, pp. 158–168, 2020.

[9] P. Chang and T. Padır, "Model-Based Manipulation of Linear Flexible Objects: Task Automation in Simulation and Real World," *Machines*, vol. 8, 2020.

[10] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking Deformable Objects with Point Clouds," *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 1130–1137, 2013.

[11] T. Tang, Y. Fan, H.-C. Lin, and M. Tomizuka, "State Estimation for Deformable Objects by Point Registration and Dynamic Simulation," in *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2017, pp. 2427–2433.

[12] T. Tang and M. Tomizuka, "Track Deformable Objects from Point Clouds with Structure Preserved Registration," *Int. J. Robot. Res.*, vol. 41, no. 6, pp. 599–614, 2022.

[13] C. Chi and D. Berenson, "Occlusion-Robust Deformable Object Tracking Without Physics Simulation," in *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2019, pp. 6443–6450.

[14] Y. Wang, D. McConachie, and D. Berenson, "Tracking Partially-Occluded Deformable Objects while Enforcing Geometric Constraints," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 14 199–14 205.

[15] Y. Yang, J. A. Stork, and T. Stoyanov, "Particle Filters in Latent Space for Robust Deformable Linear Object Tracking," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 12 577–12 584, 2022.

[16] A. L. Yuille and N. M. Grzywacz, "A Mathematical Analysis of the Motion Coherence Theory," *Int. J. Comput. Vis.*, vol. 3, no. 2, pp. 155–175, 1989.

[17] A. Myronenko and X. Song, "Point Set Registration: Coherent Point Drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, 2010.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the *EM* Algorithm," *J. R. Stat. Soc., Ser. B, Methodol.*, vol. 39, pp. 1–22, 1977.

[19] S. Ge, G. Fan, and M. Ding, "Non-Rigid Point Set Registration with Global-Local Topology Preservation," *IEEE/CVF Int. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, pp. 245–251, 2014.

[20] K. Lv, M. Yu, Y. Pu, X. Jiang, G. Huang, and X. Li, "Learning to Estimate 3-D States of Deformable Linear Objects from Single-Frame Occluded Point Clouds," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7119–7125.

[21] A. Keipour, M. Bandari, and S. Schaal, "Deformable One-Dimensional Object Detection for Routing and Manipulation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4329–4336, 2022.

[22] Z. Chen and S. Haykin, "On Different Facets of Regularization Theory," *Neural Comput.*, vol. 14, no. 12, pp. 2791–2846, 2002.

[23] A. J. Smola and B. Schölkopf, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[24] R. Zanella, A. Caporali, K. Tadaka, D. De Gregorio, and G. Palli, "Auto-Generated Wires Dataset for Semantic Segmentation with Domain Independence," in *IEEE Int. Conf. Comput. Cont. Robot. (ICCCR)*. IEEE, Jan. 2021, pp. 292–298.

[25] A. Caporali, K. Galassi, R. Zanella, and G. Palli, "FASTDLO: Fast Deformable Linear Objects Instance Segmentation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9075–9082, 2022.

[26] H. Dinkel, J. Xiang, H. Zhao, B. Coltin, T. Smith, and T. Bretl, "Wire Point Cloud Instance Segmentation from RGBD Imagery with Mask R-CNN," in *IEEE Int. Conf. Robot. Autom. (ICRA) Workshop on Representing and Manipulating Deformable Objects*, May 2022.

[27] J. Xiang and H. Dinkel, "Simultaneous Shape Tracking of Multiple Deformable Linear Objects with Global-Local Topology Preservation," in *IEEE Int. Conf. Robot. Autom. (ICRA) Workshop on Representing and Manipulating Deformable Objects*, May 2023.

[28] M. Yu, K. Lv, C. Wang, M. Tomizuka, and X. Li, "A Coarse-to-Fine Framework for Dual-Arm Manipulation of Deformable Linear Objects with Whole-Body Obstacle Avoidance," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023.