Tracking Deformable Linear Objects in RGB-D Imagery with Geodesic-Based Bayesian Coherent Point Drift

Jingyi Xiang

CS 498 Machine Perception Spring 2023 Final Project Report

Abstract

Many methods exist to track the shape of deformable linear objects given a continuous stream of RGB-D frames. However, these methods often fail to output a reasonable shape estimate when the tracked object is partially occluded. This project investigates the potential approach of imputing the displacement field for the occluded portion of the deformable object. The displacement field imputation is achieved by extending Geodesic-Based Coherent Point Drift to incorporate correspondence priors.

1. Introduction

This project implements and extends the Geodesic-Based Bayesian Coherent Point Drift (GBCPD) algorithm for real-time tracking of deformable linear object (DLO) (e.g., rope, wire, string) shapes [11]. Monitoring the shape of deformable objects is essential to manipulation tasks such as knot tying or wire routing, or to monitor wires and cables for collision prevention [13,15,27–30]. These canonical tasks are common in applications like robotic surgery, industrial automation, power line avoidance, and human habitat maintenance [3, 8, 14, 16, 25]. Deformable object shape tracking is challenging in this context because cloth, ropes, and cables are often featureless and prone to occlusion and self-occlusion. Instead of incorporating physics simulators for DLO tracking, which could be computationally expensive and difficult to tune, this project investigates imputing the displacement of the occluded portion of the object based on the displacement of the visible portion of the object as a potential approach. In summary, this project achieves the following:

1. This project implements Geodesic-Based Bayesian Coherent Point Drift and integrates it with our previous work on DLO shape tracking, the TrackDLO algorithm (under review).
2. This project extends Geodesic-Based Bayesian Coherent Point Drift to take account into correspondence priors (only partially completed at the time of submission, see details in Sec.3.3).
3. The performance of the proposed method is evaluated and compared to that of the existing methods.
4. The code and data used in this project are available at https://github.com/jingyi-xiang/bcpd-dlo-tracking.

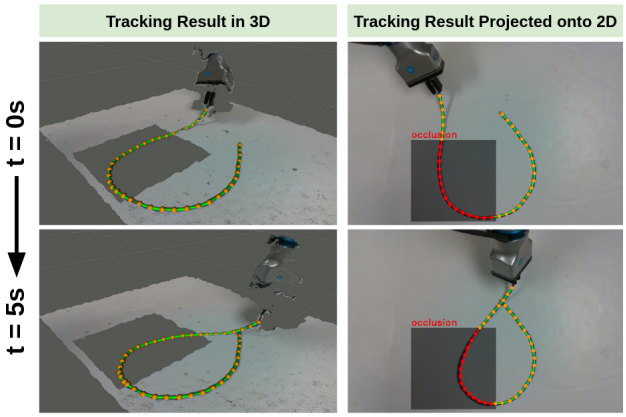

Figure 1. Geodesic-Based Coherent Point Drift combined with our previous work, the TrackDLO algorithm, accurately tracks the shape of a DLO moving under occlusion. In the left image, all point cloud in the occluded region is removed.

2. 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 [5, 17, 18, 31].

Coherent Point Drift 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 engine to update the ob-

ject's shape estimate [21]. Building on top of CPD+Physics, Structure Preserved Registration (SPR) adds in modified locally linear embedding for more accurate tracking results. Another algorithm is Constrained Deformable CPD (CDCPD), which uses locally linear embedding combined with CPD for non-rigid registration, enforces an object stretching constraint, and detects and recovers from tracking failure [2]. The CDCPD2 algorithm builds on top of CDCPD and incorporates the diminishing rigidity DLO physics model, self-intersection constraints, and obstacle interaction constraints to improve tracking under tip occlusion and large-scale occlusion [26]. Recently, we proposed the TrackDLO algorithm (under review), which exploits the motion coherence regularization embedded in CPD to impute the displacement of the occluded portion of the DLO from the displacement of the visible portion of the DLO. Our method demonstrated more robustness under various types of occlusions compared to existing methods.

Since the release of the initial CPD paper in 2006, there have been many subsequent non-rigid registration papers that improve CPD's registration accuracy and computation speed, one being Geodesic-Based Bayesian Coherent Point Drift (GBCPD) [11]. The GBCPD algorithm's various modifications to the original CPD algorithm inspired this project to investigate its performance when applied to DLO tracking. Our previous work, the TrackDLO algorithm, consists of two parts: a pre-processing step followed by a CPD-based registration. This project replaces the CPD-based registration in TrackDLO with a GBCPD-based one.

3. Methods

3.1. Problem Formulation

For N *points* of dimension D received at time t from a depth sensor, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the DLO shape is represented with a collection of M ordered *nodes*, $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$ and the $M-1$ edges connecting the adjacent nodes. The measurement X can contain outliers due to noise and it can be incomplete due to occlusion. Usually $M \ll N$. The objective of DLO shape tracking is to estimate Y given the current measurement of X and the previous estimation of Y. Many existing literature treats DLO shape tracking as a non-rigid registration problem [4, 21–23, 26]. The rest of the methods section is structured as follows: Sec.3.2 gives an overview of the Geodesic-Based Bayesian Coherent Point Drift algorithm [11], Sec.3.3 goes over our modifications to GBCPD which makes it compatible with the TrackDLO algorithm, and Sec.3.4 discusses some of the implementation details.

3.2. Geodesic-Based Bayesian Coherent Point Drift

The GBCPD algorithm computes the alignment between X and Y through variational inference. It considers Y (the source point set) as the centroids of a Gaussian mixture model (GMM) and X (the target point set) as the data points to be fitted to. Besides estimating the locations of Y, the correct correspondence between X and Y also needs to be computed. Let $\mathcal{T}$ be a function that deforms Y to match with the shape of X, GBCPD defines $\mathcal{T}$ as a combination of similarity and non-rigid transformation:

$$\mathcal{T}(\mathbf{y}_m) = s\mathbf{R}(\mathbf{y}_m + \mathbf{v}_m) + \mathbf{t} \tag{1}$$

where s is a scale factor, $\mathbf{R}$ is a rotation matrix, $\mathbf{t}$ is a translation vector, and $\mathbf{v}$ is a displacement field representing the non-rigid transformation with $\mathbf{v}_m$ being the displacement vector for $\mathbf{y}_m$. If $\mathbf{x}_n$ corresponds to $\mathbf{y}_m$, $\mathbf{x}_n$ is generated from a Gaussian distribution with covariance matrix $\sigma^2\mathbf{I}_D$ centered at $\mathcal{T}(\mathbf{y}_m)$. The probability of $\mathbf{x}_n$ being an outlier is ω and outliers are generated from a distribution p_{out}. To compute the transformation $\mathcal{T}$ and the correspondence between the source point set Y and the target point set X, GBCPD defines the following notations:

- $\mathbf{x}_{DN\times 1} = (\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T)^T$ – the vector representation of the target point set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.
- $\mathbf{y}_{DM\times 1} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_M^T)^T$ – the vector representation of the source point set $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$.
- $\mathbf{v}_{DM\times 1} = (\mathbf{v}_1^T, \ldots, \mathbf{v}_M^T)^T$ – the vector representation of the displacement field that deforms Y.
- $c_n \in \{0, 1\}$ – an indicator variable where $c_n = 1$ only if the $\mathbf{x}_n$ is not an outlier.
- $e_n \in \{0, \ldots, M\}^N$ – an index variable representing that $\mathbf{x}_n$ corresponds to $\mathbf{y}_m$ if $e_n = m$.
- $\alpha_m \in [0, 1]^M$ – the probability of event $e_n = m$ for any n which satisfies $\sum_{m=1}^{M} \alpha_m = 1$.
- $\phi(\mathbf{z}; \boldsymbol{\mu}, \mathbf{S})$ – a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance $\mathbf{S}$.
- $\rho = (s, \mathbf{R}, \mathbf{t})$ – the similarity transformation parameters.

The joint distribution of $(\mathbf{x}_n, c_n, e_n)$ given $(\mathbf{y}, \mathbf{v}, \alpha, \rho, \sigma^2)$ can then be modeled as

$$\begin{aligned} &p(\mathbf{x}_n, c_n, e_n | \mathbf{y}, \mathbf{v}, \alpha, \rho, \sigma^2) \\ &= \{\omega p_{\text{out}}(\mathbf{x}_n)\}^{1-c_n} \left\{ (1-\omega) \prod_{m=1}^{M} (\alpha_m \phi_{mn})^{\delta_m(e_n)} \right\}^{c_n} \end{aligned} \tag{2}$$

where $\delta_m(e_n) = 1$ if $e_n = m$ and 0 otherwise. Additionally, ϕ_{mn} is the abbreviation of a Gaussian distribution that takes the form

$$\begin{aligned} \phi_{mn} &= \phi(\mathbf{x}_n; \mathbf{y}_m, \sigma^2\mathbf{I}_D) \\ &= |2\pi\sigma^2\mathbf{I}_D|^{-1/2} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_n - \mathcal{T}(\mathbf{y}_m)\|^2 \right) \end{aligned} \tag{3}$$

Without posing any constraints on the displacement field $\mathbf{v}$, the problem is ill-posed. One method to resolve this is

to require the displacement field to be as smooth as possible, as stated in the Motion Coherence Theory [31]. Unlike the original Coherent Point Drift algorithm which incorporates motion coherence as a regularization term, GBCPD models motion coherence with a prior distribution of $\mathbf{v}$. Suppose $\mathbf{G}$ is a positive definite matrix with its elements $\mathbf{G}(i,j) = \mathcal{K}(\mathbf{y}_i, \mathbf{y}_j)$ and $\mathcal{K}$ is a positive definite kernel, the prior distribution is formulated as follows:

$$p(\mathbf{v}|\mathbf{y}) = \phi(\mathbf{v}; 0, \lambda^{-1}\mathbf{G} \otimes \mathbf{I}_D) \tag{4}$$

where $\otimes$ is the Kronecker product. The Motion Coherence Theory states that features close to each other in space are more likely to move in similar directions and speeds; this prior distribution encodes that since $\mathbf{v}_i$ and $\mathbf{v}_j$ correlates with each other if $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j)$ is large enough. The GBCPD algorithm further assumes $p(\alpha)$ is a Dirichlet distribution characterized by a parameter κ:

$$p(\alpha) = \mathrm{Dir}(\alpha|\kappa\mathbf{1}_M) \tag{5}$$

where $\mathbf{1}_M$ is a column vector of 1 s.

Denote the set of latent variables that need to be estimated as $\theta = (\mathbf{v}, \alpha, \mathbf{c}, \mathbf{e}, s, \mathbf{R}, \mathbf{t}, \sigma^2)$, GBCPD defines the full joint distribution $p(\mathbf{x}, \mathbf{y}, \theta)$ as

$$p(\mathbf{x}, \mathbf{y}, \theta) \propto p(\mathbf{v}|\mathbf{y})p(\alpha)\prod_{n=1}^{N} p(\mathbf{x}_n, c_n, e_n|\mathbf{y}, \alpha, \rho, \sigma^2) \tag{6}$$

The GBCPD algorithm then estimates a set of reasonable θ using variational inference. The posterior distribution $p(\theta|\mathbf{x}, \mathbf{y})$ is approximated with a distribution $q(\theta)$ through which the expectation can be easily computed:

$$q(\theta) = q_1(\mathbf{v}, \alpha)q_2(\mathbf{c}, \mathbf{e})q_3(\rho, \sigma^2) \tag{7}$$

After initialization, q_1, q_2, and q_3 are updated one at a time with the other two q fixed. The update step is repeated until $q(\theta)$ converges. The GBCPD algorithm defines the following notations for the update equations:

- $\mathbf{P}_{M\times N}$ – the posterior probability matrix where $\mathbf{P}(m,n)$ represents the probability that $\mathbf{x}_n$ corresponds to $\mathbf{y}_m$.
- $\boldsymbol{\nu}_{M\times 1} = (\nu_1, \ldots, \nu_M)^T$ with $\nu_m = \sum_{n=1}^{N}\mathbf{P}(m,n)$ – the estimated number of target points matched with each source point.
- $\boldsymbol{\nu}'_{N\times 1} = (\nu'_1, \ldots, \nu'_N)^T$ with $\nu_n = \sum_{m=1}^{M}\mathbf{P}(m,n)$ – the probability that $\mathbf{x}_n$ is a non-outlier.
- $\hat{N} = \sum_{n=1}^{N}\sum_{m=1}^{M}\mathbf{P}(m,n)$ – the estimated number of points in X that match to Y.
- $\tilde{\mathbf{M}} = \mathbf{M} \otimes \mathbf{1}_D$ – tilde symbol on top of a matrix indicates the Kronecker product of itself with a column vector of 1 s.
- ψ – the digamma function.

The update of $q_1(\mathbf{v}, \alpha)$ updates the following variables:

$$\alpha_m = \exp(\psi(\kappa + \nu_m) - \psi(\kappa M + \hat{N})) \tag{8}$$

$$\boldsymbol{\Sigma}^{-1} = \lambda\mathbf{G}^{-1} + \frac{s^2}{\sigma^2}\mathrm{d}(\boldsymbol{\nu}) \tag{9}$$

$$\hat{\mathbf{x}} = \mathrm{d}(\tilde{\boldsymbol{\nu}})^{-1}\tilde{\mathbf{P}}\mathbf{x} \tag{10}$$

$$\hat{\mathbf{v}} = \frac{s^2}{\sigma^2}\tilde{\boldsymbol{\Sigma}}\mathrm{d}(\tilde{\boldsymbol{\nu}})(\tilde{\mathbf{T}}^{-1}(\hat{\mathbf{x}}) - \mathbf{y}) \tag{11}$$

$$\hat{\mathbf{u}} = y + \hat{\mathbf{v}} \tag{12}$$

$$\hat{\mathbf{y}} = s(\mathbf{I}_M \otimes \mathbf{R})(\mathbf{y} + \hat{\mathbf{v}}) + \mathbf{1}_M \otimes \mathbf{t} \tag{13}$$

The update of $q_2(\mathbf{c}, \mathbf{e})$ updates the posterior probability matrix $\mathbf{P}$:

$$\langle\phi_{mn}\rangle = \phi(\mathbf{x}_n; \hat{\mathbf{y}}_m, \sigma^2\mathbf{I}_D)\exp(-\frac{s^2 D}{2\sigma^2}\boldsymbol{\Sigma}(m,m)) \tag{14}$$

$$\mathbf{P}(m,n) = \frac{(1-\omega)\alpha_m\langle\phi_{mn}\rangle}{\omega p_{\mathrm{out}}(\mathbf{x}_n) + (1-\omega)\sum_{m'=1}^{M}\alpha_{m'}\langle\phi_{m'n}\rangle} \tag{15}$$

The update of $q_3(\phi, \sigma^2)$ updates s, $\mathbf{R}$, $\mathbf{t}$, and σ^2:

$$\hat{\mathbf{R}} = \boldsymbol{\Phi}\mathrm{d}(1, \ldots, 1, |\boldsymbol{\Phi}\boldsymbol{\Psi}^T|)\boldsymbol{\Psi}^T \tag{16}$$

$$\hat{s} = \mathrm{tr}(\hat{\mathbf{R}}^T\mathbf{S}_{xu})/\mathrm{tr}(\mathbf{S}_{uu}) \tag{17}$$

$$\hat{\mathbf{t}} = \bar{\mathbf{x}} - \hat{s}\hat{\mathbf{R}}\bar{\mathbf{u}} \tag{18}$$

$$\hat{\sigma}^2 = \frac{1}{\hat{N}D}(\mathbf{x}^T\mathrm{d}(\tilde{\boldsymbol{\nu}}')\mathbf{x} - 2\mathbf{x}^T\tilde{\mathbf{P}}^T\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathrm{d}(\tilde{\boldsymbol{\nu}})\hat{\mathbf{y}}) + \hat{s}^2\bar{\sigma}^2 \tag{19}$$

where

$$\bar{\mathbf{x}} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\hat{\mathbf{x}}_m,\ \bar{\mathbf{u}} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\hat{\mathbf{u}}_m \tag{20}$$

$$\bar{\sigma}^2 = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m\boldsymbol{\Sigma}(m,m)^2 \tag{21}$$

$$\mathbf{S}_{xu} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m(\hat{\mathbf{x}}_m - \bar{\mathbf{x}})(\hat{\mathbf{u}}_m - \bar{\mathbf{u}})^T \tag{22}$$

$$\mathbf{S}_{uu} = \frac{1}{\hat{N}}\sum_{m=1}^{M}\nu_m(\hat{\mathbf{u}}_m - \bar{\mathbf{u}})(\hat{\mathbf{u}}_m - \bar{\mathbf{u}})^T + \bar{\sigma}^2\mathbf{I}_D \tag{23}$$

The final output of GBCPD is $\hat{\mathbf{y}} = s(\mathbf{I}_M \otimes \mathbf{R})(\mathbf{y} + \hat{\mathbf{v}}) + \mathbf{1}_M \otimes \mathbf{t}$.

3.3. Incorporating Correspondence Priors

When tracking the shape of a DLO, the point cloud X received at each frame is often incomplete due to occlusion. This occlusion can be caused by robotic manipulators, other objects in the scene, or the DLO itself. The occluded nodes in Y often do not contribute to generating points in

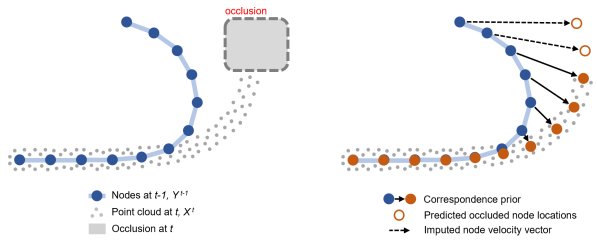

Figure 2. By "pinning" visible nodes in Y^{t-1} onto their known correspondences in X^t (solid line arrows), the displacement field of the occluded node in Y^{t-1} can be imputed (dashed line arrows).

X. In this case, the displacement of the occluded nodes is mostly regulated by the Motion Coherence Theory which correlates their displacement with that of their neighboring visible nodes.

To help address the DLO tracking under occlusion problem, we assume that the DLO being tracked can deform but cannot be stretched or compressed. As a result, its total length and segment lengths (distance between adjacent nodes) should remain unchanged during deformation. In our previous work, TrackDLO, we proposed a preprocessing step that estimates the new locations of the visible nodes such that the segment distances are preserved if the visible nodes are aligned to these estimated locations during the registration step. Since motion coherence is encoded in CPD and GBCPD, the displacement of the occluded nodes will be imputed from the displacement of the visible nodes if the visible nodes are "pinned" to these estimated locations. This concept is showcased in Fig.2. To save space, we will not go into detail about this preprocessing process.

Implementing the above pipeline requires modification to the original GBCPD algorithm so it takes into account that selected nodes in Y should be aligned with selected points in X. Suppose we have pre-computed the alignment of the visible nodes. Let $(\mathbf{x}_a, \mathbf{y}_b)$ be the pre-computed correspondence priors and N_c be the collection of indices (a, b) of all correspondence priors. To incorporate these known alignments with desired displacement field, we follow the formulation in Extended Coherent Point Drift [6] and model this as a product of independent density functions:

$$P_c(N_c) = \prod_{(a,b)\in N_c} |2\pi\zeta \mathbf{I}_D|^{-1/2} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_a - \mathcal{T}(\mathbf{y}_b)\|^2\right) \quad (24)$$

where the parameter ζ indicates the priors' degree of reliability. The smaller ζ is, the closer $\mathbf{x}_a$ and $\mathbf{y}_b$ will be aligned in the final registration result. We add $P_c(N_c)$ to (6) and

Algorithm 1 GBCPD with Correspondence Priors

Input: X, Y, N_c
$X_c \leftarrow$ the collection of all $\mathbf{x}_a$ that satisfies $(a, \cdot) \in N_c$
$Y_c \leftarrow$ the collection of all $\mathbf{y}_b$ that satisfies $(\cdot, b) \in N_c$
$s, \mathbf{R}, \mathbf{t} \leftarrow$ GBCPD $(X = X_c, Y = Y_c)$
$X' \leftarrow \mathbf{R}^{-1}(X - \mathbf{t})/s$
$X'_c \leftarrow \mathbf{R}^{-1}(X_c - \mathbf{t})/s$
$N'_c \leftarrow$ new corr. priors constructed with X'_c and Y_c
$Y_{\text{new}} \leftarrow$ Extended GBCPD
$(X = X', Y = Y, N_c = N'_c)$
$Y_{\text{new}} \leftarrow s\mathbf{R}(Y_{\text{new}}) + \mathbf{t}$
Return Y_{new}

Algorithm 2 TrackDLO + GBCPD

while detecting target DLO in RGB-D stream **do**
  $X^t \leftarrow$ segment and downsample DLO pointcloud
  $Y_c^t \leftarrow$ the visible nodes in Y^{t-1}, $Y_c^t \subseteq Y^{t-1}$
  $X_c^t \leftarrow$ estimated locations of Y_c^t such that segment distances are preserved
  $X^t \leftarrow X^t \cup X_c^t$
  $N_c \leftarrow$ compute corr. priors between X^t and Y^{t-1} such that X_c^t corresponds to Y_c^t
  $Y^t \leftarrow$ GBCPD with correspondence priors $(X = X^t, Y = Y^{t-1}, N_c = N_c)$
end while

obtain

$$p(\mathbf{x}, \mathbf{y}, \theta) \propto p(\mathbf{v}|\mathbf{y})p(\alpha)P_c(N_c)\prod_{n=1}^{N} p(\mathbf{x}_n, c_n, e_n|\mathbf{y}, \alpha, \rho, \sigma^2) \quad (25)$$

This modification affects the GBCPD update equations for q_1 and q_3. We follow the same derivation process in that of GBCPD to derive the new update equation for q_1 [10]. Here we directly give the results. Let matrix $\mathbf{J}_{M\times N}$ be a matrix encoding the correspondence between X and Y, where $\mathbf{J}(a, b) = 1$ if $(a, b) \in N_c$ and 0 otherwise. We also define a new variable $\boldsymbol{\nu}_c = \mathbf{J}\mathbf{1}_N$ analogous to $\boldsymbol{\nu}$. The new update equations for $\boldsymbol{\Sigma}$ and $\hat{\mathbf{v}}$ are

$$\boldsymbol{\Sigma}^{-1} = \lambda\mathbf{G}^{-1} + \frac{s^2}{\sigma^2}\mathrm{d}(\boldsymbol{\nu}) + \frac{s^2}{\zeta}\mathrm{d}(\boldsymbol{\nu}_c) \quad (26)$$

$$\hat{\mathbf{v}} = \frac{s^2}{\sigma^2}\tilde{\boldsymbol{\Sigma}}\mathrm{d}(\tilde{\boldsymbol{\nu}})(\tilde{\mathbf{T}}^{-1}(\hat{\mathbf{x}}) - \mathbf{y}) + \frac{s^2}{\zeta}\tilde{\boldsymbol{\Sigma}}\mathrm{d}(\tilde{\boldsymbol{\nu}}_c)(\tilde{\mathbf{T}}^{-1}(\tilde{\mathbf{J}}\mathbf{x}) - \mathbf{y}) \quad (27)$$

Due to the time constraint on this project, we were only able to derive the new update equation for q_1 in time. Without proper update equations for q_3 and therefore s, $\mathbf{R}$, and $\mathbf{t}$, the similarity transformation in (1) is ignored altogether

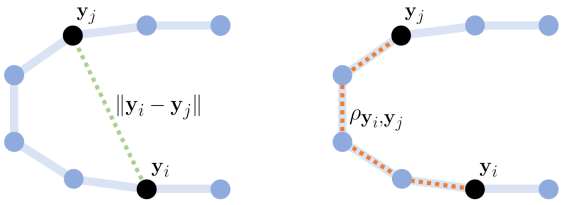

Figure 3. The geodesic distance $\rho_{i,j}$ (orange, right) better represents the distance between nodes describing the shape of a DLO than the Euclidean distance $\|\mathbf{y}_m^t, \mathbf{y}_i^t\|$ (green, left).

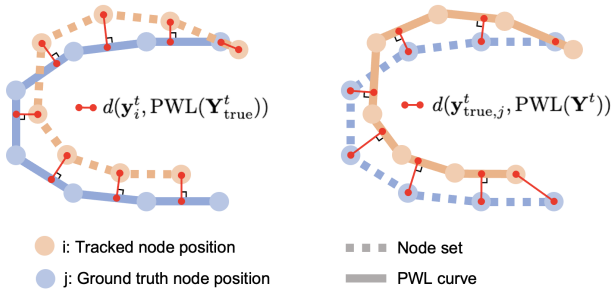

Figure 4. The frame error is the average of the errors from the left and right images. (Left) Node-to-PWL curve $\varepsilon(Y^t, Y_{\text{true}}^t)$. (Right) Node-to-PWL curve $\varepsilon(Y_{\text{true}}^t, Y^t)$.

and $\mathcal{T}(\mathbf{y}_m)$ reduces to $\mathbf{y}_m + \mathbf{v}_m$. We still estimate the similarity transformation parameters by applying GBCPD twice per registration, once including the similarity transformation but not the correspondence priors and once including the correspondence priors but not the similar transformation. The process is described in detail in Alg.1. For readability, we temporarily refer to our modified GBCPD algorithm as Extended GBCPD, analogous to Extended Coherent Point Drift. Extended GBCPD takes into account correspondence priors but has fixed $s = 1$, $\mathbf{R} = \mathbf{I}_D$, and $\mathbf{t} = \mathbf{0}$.

The intuition of Alg.1 is that after computing s, $\mathbf{R}$, and $\mathbf{t}$ between all points in Y that have correspondences and their correspondences in X, we reverse the similarity transformation to obtain the displacement field of points in Y that have correspondences (visible nodes in the context of DLO tracking). We can then use Extended GBCPD to impute the displacement field of points in Y that do not have correspondences (occluded nodes in the context of DLO tracking). Finally, this imputed displacement field, combined with the transformation parameters computed from the first GBCPD registration, gives the final new locations of Y using (1).

The full DLO tracking pipeline that utilizes GBCPD with correspondence priors is described in Alg.2. The final registration step in line 9, which uses CPD with correspondence priors in our previous work, now uses the proposed modified GBCPD algorithm with correspondence priors.

3.4. Kernel used for Motion Coherence

The kernel $\mathcal{K}$ used in the motion coherence matrix $\mathbf{G}$ directly affects the displacement field imputation results. In this project, we use a kernel of the following form:

$$\begin{aligned}\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \; & 0.005 e^{-\|\rho_{i,j}\|/(2\beta^2)} \\ & + 0.995 \frac{1}{2\beta^2} e^{-2\|\rho_{i,j}\|/\beta} (2\|\rho_{i,j}\| + \beta)\end{aligned} \tag{28}$$

where $\rho_{i,j}$ is the approximated geodesic distance between $\mathbf{y}_i$ and $\mathbf{y}_j$. As shown in Fig.3, $\rho_{i,j}$ is defined as:

$$\rho_{\mathbf{y}_i,\mathbf{y}_j} = \begin{cases} \sum_{m=j}^{i-1} \|\mathbf{y}_{m+1} - \mathbf{y}_m\| & \text{if } j \leq i \\ \sum_{m=i}^{j-1} \|\mathbf{y}_{m+1} - \mathbf{y}_m\| & \text{if } j > i \end{cases}. \tag{29}$$

The above kernel is a weighted sum of the Laplacian kernel and the kernel used in TrackDLO. Currently, the above kernel form is selected based on intuition and limited testing. Further study is required to unveil the connection between different kernel forms and how they affect the motion coherence between nodes.

4. Experiments

We conducted experiments to compare the performance of TrackDLO+GBCPD, TrackDLO, CDCPD2 with and without optional gripper information, and CDPCD algorithms for DLO tracking under occlusion. Among the four algorithms tested in three scenarios, TrackDLO+GBCPD demonstrated the lowest node-to-PWL curve frame error in two out of the three scenarios (Sec.4.1). We also analyze the effect of the similarity transformation in GBCPD on tracking accuracy (Sec.4.2). For all experiments below, we used $\lambda = 40$, $\omega = 0$, $\kappa = 1 \times 10^{16}$, $\zeta = 1 \times 10^{-8}$, and convergence tolerance = 0.0001. We initialized σ^2 with the σ^2 from the last time step multiplied by a scale factor $\gamma = 1.2$. For experiments in Sec.4.1 where a long rope was used, we used $\beta = 5$. For experiments in Sec.4.2 where a short rope was used, we used $\beta = 2$. We also fix the scale s at 1 in GBCPD.

4.1. Tracking Error

We conducted experiments to compare the performance of TrackDLO+GBCPD, TrackDLO, CDCPD2 with and without optional gripper information, and CDPCD algorithms in three different scenarios. These scenarios include:

1. *Stationary*–A curved rope lies stationary on a table. This scenario tests tracking error accumulation and length preservation under tip occlusion.
2. *Perpendicular Motion*–A robot arm with a gripper moves the tip of a curved rope in a direction perpendicular to the tip while the mid-section is occluded. This scenario tests tracking accuracy when both tips of the rope are visible while the mid-section is occluded.

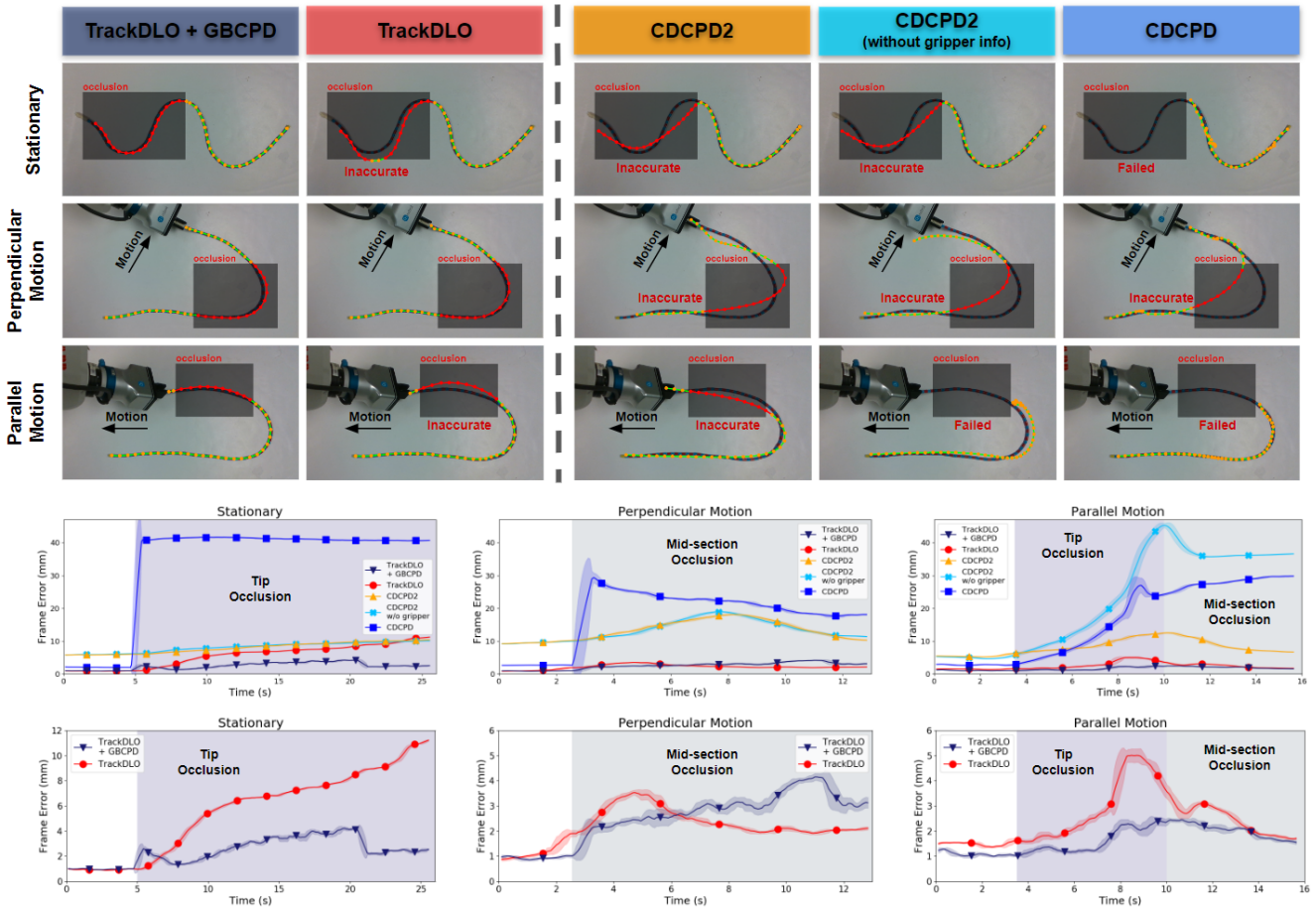

Figure 5. TrackDLO+GBCPD accurately estimates the state of the DLO under scaled, tip, and mid-section occlusion in the three evaluation scenarios as compared to the original TrackDLO, CDCPD2 with and without gripper information, and CDCPD. Among the algorithms evaluated, TrackDLO+GBCPD had the lowest frame error in the *Stationary* and *Parallel Motion* case, while TrackDLO had the lowest frame error in the *Perpendicular Motion* case. The background of the plots are colored so that light purple indicates tip occlusion and light gray indicates mid-section occlusion.

3. *Parallel Motion*–A robot arm with a gripper moves the tip of the rope through occlusion in a direction parallel to the tip. This scenario tests length preservation and tracking accuracy when one tip of the rope moves through occlusion.

For each scenario, point cloud and RGB image data were collected and saved in a Robot Operating System (ROS) bag file which was used for evaluation. Occlusion was injected by removing pixels within a bounding box area in the DLO segmentation mask. The blue rope 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 detected the blue and red segments along the rope, and keypoint detection returned their centroids. These L centroids were combined to form Y^t_{true}, the ground truth DLO nodes, which were compared to the M track nodes in Y^t. Note the markers on the rope were only used for evaluation purposes and they are not required by any algorithm for tracking. Each experiment was repeated 10 times for each algorithm in each scenario.

Tracking performance is evaluated using a frame error metric, defined as follows and depicted in Fig.4. The set of points $\text{PWL}(Y)$ in the Piecewise Linear (PWL) curve representation of an ordered node sequence Y includes both the nodes themselves and all points in the line segments connecting consecutive nodes. We define the point-set distance

$$d(\mathbf{y}, \mathcal{S}) = \inf_{\mathbf{y}' \in \mathcal{S}} \|\mathbf{y} - \mathbf{y}'\| \tag{30}$$

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

$$\varepsilon(Y^t, Y^t_{\text{true}}) = \frac{1}{M} \sum_{\mathbf{y}^t_i \in \mathbf{Y}^t} d(\mathbf{y}^t_i, \text{PWL}(Y^t_{\text{true}})) \tag{31}$$

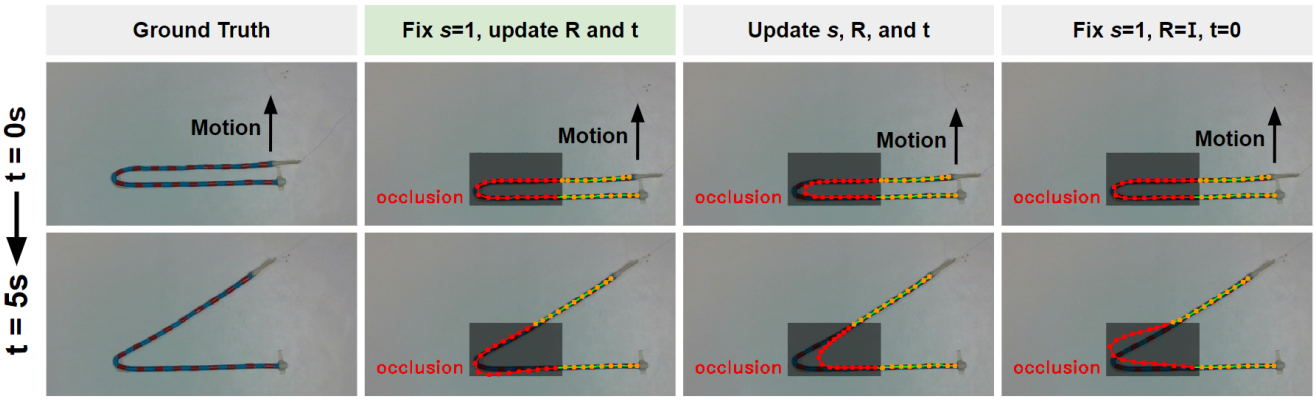

Figure 6. TrackDLO+GBCPD with rigid transformation (fix $s = 1$, update $\mathbf{R}$ and $\mathbf{t}$) demonstrated the most accurate displacement field imputation result. When similarity transformation is used instead, the tracking result shrinks in length, potentially because the scale s is not fixed at 1. When no transformation is included, the tracking result is not smooth and contains kinks.

and the frame error metric mirrors this per-node error to guarantee symmetry:

$$\mathcal{E}(Y^t_{\text{true}}, Y^t) = \frac{1}{2}\left(\varepsilon(Y^t_{\text{true}}, Y^t) + \varepsilon(Y^t, Y^t_{\text{true}})\right) \quad (32)$$

Evaluation results are reported in Figure 5. In the *Stationary* scenario, occlusion is injected at $t = 5s$ and half of the DLO is then occluded for 20 seconds. The TrackDLO+GBCPD algorithm achieves the lowest average frame error for the *Stationary* scenario. In the *Perpendicular Motion* scenario, occlusion is injected in the mid-section of the rope three seconds after tracking begins before the rope begins to move and remains there until the end of the bag file. For the CDCPD2 and CDCPD algorithms, the error increases after the injection of occlusion and decreases as the tracking estimates begin to catch up with the wire state. The TrackDLO algorithm achieves the lowest average frame error in this scenario. In the *Parallel Motion* scenario, the rope begins in an unoccluded state and the gripper moves the rope tip through an occluded region. Tracking error increases for all algorithms until the rope tip becomes visible again during the mid-section occlusion period. The TrackDLO+GBCPD algorithm achieves the lowest frame error in this scenario.

One interesting phenomenon worth noting is the two error drops at $t = 5.5s$ and $t = 20s$ for TrackDLO+GBCPD in the *Stationary* scenario. All other algorithms in this scenario, except for CDCPD which failed completely, had monotonically increasing errors as functions of time. This is because these algorithms update the location of nodes with $\mathbf{y}^t_m = \mathbf{y}^{t-1}_m + \mathbf{v}_m$; once the error accumulates it is irreversible. The similarity transformation parameters estimated in GBCPD, which do not depend on their previous estimated values, help correct the accumulated errors in Y.

4.2. Rigid vs. Similarity Transformation

We conducted experiments to investigate the effect of the similarity transformation in GBCPD's deformation model. We ran TrackDLO+GBCPD on three different settings:

1. Fix $s = 1$, update $\mathbf{R}$ and $\mathbf{t}$ – this setup reduces the similarity transformation to a rigid transformation.
2. Update s, $\mathbf{R}$, and $\mathbf{t}$ – this is the original setup in GBCPD.
3. Fix $s = 1, \mathbf{R} = \mathbf{I}_D, \mathbf{t} = \mathbf{0}$ – this setup ignores the similarity transformation completely and the deformation model reduces to $\mathbf{y}^t_m = \mathbf{y}^{t-1}_m + \mathbf{v}_m$.

For the experiment setup, we used the following scenario: a short rope is initially folded and one half of the rope is moved upwards, away from the other half. Occlusion is injected over the mid-section of the rope once it begins to move. This scenario tests the accuracy of displacement field imputation for occluded nodes produced by the three above settings, while the DLO is moving.

The experiment results are reported in Fig.6. When the scale s in (1) is fixed at 1 and therefore rigid transformation is used, the imputed displacement field is the most accurate. When similarity transformation is used instead, the tracking result shrinks in length, potentially because the scale s can drop below 1 and therefore shrinks the object size. When no transformation is included, the tracking result is not smooth and contains kinks. The results show the importance of including the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ in the deformation model. Since we assume the DLO has fixed total and segment lengths, it is reasonable to fix the scale at 1.

5. Conclusions and Future Work

This project extends the Geodesic-Based Bayesian Coherent Point Drift algorithm and integrates it with our previ-

ous work, TrackDLO. The GBCPD algorithm demonstrated potential for being used in DLO shape tracking as the combination of TrackDLO and GBCPD leads to improved results in the failure cases of the original TrackDLO algorithm. The math principles behind GBCPD still need to be studied rigorously for deriving the closed-form similarity transformation parameter update equations when correspondence priors are incorporated. Future work could also investigate how different kernels affect motion coherence.

Acknowledgements

The author thanks the Bretl Research Group and the Representing and Manipulating Deformable Linear Objects project at UIUC for providing lab equipment, as well as the co-authors of the TrackDLO paper for developing the evaluation framework used in this project. The author also thanks the teams developing and maintaining the open-source software used in this project, including ROS, OpenCV, NumPy, Matplotlib, Open3D, SciPy, Point Cloud Library, and Eigen3 [1, 7, 9, 12, 19, 20, 24, 32].

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 8

[2] C. Chi and C. Berenson. Occlusion-Robust Deformable Object Tracking Without Physics Simulation. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2019. 2

[3] Peng Chang and Taşkın Padır. Model-Based Manipulation of Linear Flexible Objects: Task Automation in Simulation and Real World. *Machines*, 8, 2020. 1

[4] Cheng Chi and Dmitry Berenson. Occlusion-Robust Deformable Object Tracking Without Physics Simulation. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, pages 6443–6450, 2019. 2

[5] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data Via the *EM* Algorithm. *J. R. Stat. Soc., Ser. B, Methodol.*, 39:1–22, 1977. 1

[6] Vladislav Golyanik, Bertram Taetz, Gerd Reis, and Didier Stricker. Extended Coherent Point Drift Algorithm with Correspondence Priors and Optimal Subsampling, Supplementary Material. 4

[7] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. 8

[8] Jiuming Guo, Jiwen Zhang, Dan Wu, Yuhang Gai, and Ken Chen. An Algorithm Based on Bidirectional Searching and Geometric Constrained Sampling for Automatic Manipulation Planning in Aircraft Cable Assembly. *J. Manuf. Syst.*, 57:158–168, 2020. 1

[9] Charles R. Harris, Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array Programming with NumPy. *Nature*, 585:357–362, 2020. 8

[10] O. Hirose. A bayesian formulation of coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(07):2269–2286, jul 2021. 4

[11] Osamu Hirose. Geodesic-based bayesian coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 2

[12] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.*, 9(3):90–95, 2007. 8

[13] Shiyu Jin, Wenzhao Lian, Changhao Wang, Masayoshi Tomizuka, and Stefan Schaal. Robotic Cable Routing with Spatial Representation. In *IEEE Robot. Autom. Lett.*, volume 7, pages 5687–5694, Apr. 2022. 1

[14] Azarakhsh Keipour, Maryam Bandari, and Stefan Schaal. Efficient Spatial Representation and Routing of Deformable One-Dimensional Objects for Manipulation. *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, pages 211–216, 2022. 1

[15] Romain Lagneau, Alexandre Krupa, and Maud Marchal. Automatic Shape Control of Deformable Wires Based on Model-Free Visual Servoing. In *IEEE Robot. Autom. Lett.*, volume 5, pages 5252–5259, Oct. 2020. 1

[16] Bo Lu, Henry K Chu, and Li Cheng. Dynamic Trajectory Planning for Robotic Knot Tying. In *IEEE Int. Conf. Real-Time Comput. Robot. (RCAR)*, pages 180–185, 2016. 1

[17] Andriy Myronenko and Xubo Song. Point Set Registration: Coherent Point Drift. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2262–2275, 2010. 1

[18] Andriy Myronenko, Xubo Song, and Miguel Carreira-Perpiñá. Non-Rigid Point Set Registration: Coherent Point Drift. *Adv. Neur. Inf. Proc. (NeurIPS)*, pages 1–8, 2006. 1

[19] Radu Bogdan Rusu and Steve Cousins. 3D is Here: Point Cloud Library (PCL). In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1–4, 2011. 8

[20] Stanford Artificial Intelligence Laboratory. Robotic Operating System: Noetic Ninjemys, 2018. 8

[21] Te Tang, Yongxiang Fan, Hsien-Chung Lin, and Masayoshi Tomizuka. State Estimation for Deformable Objects by Point Registration and Dynamic Simulation. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, pages 2427–2433, 2017. 2

[22] Te Tang and Masayoshi Tomizuka. Track Deformable Objects from Point Clouds with Structure Preserved Registration. *Int. J. Robot. Res.*, 41(6):599–614, 2022. 2

[23] Te Tang, Changhao Wang, and Masayoshi Tomizuka. A Framework for Manipulating Deformable Linear Objects by Coherent Point Drift. *IEEE Robot. Autom. Lett.*, 3(4):3426–3433, 2018. 2

[24] Pauli Virtan, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods*, 17:261—-272, 2020. 8

[25] Vainavi Viswanath, Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Ellen Novoseller, Jeffrey Ichnowski, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. Disentangling Dense Multi-Cable Knots. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, pages 3731–3738, 2021. 1

[26] Y. Wang, D. McConachie, and D. Berenson. Tracking Partially-Occluded Deformable Objects while Enforcing Geometric Constraints. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 14199–14205, 2021. 2

[27] Mengyuan Yan, Gen Li, Yilin Zhu, and Jeannette Bohg. Learning Topological Motion Primitives for Knot Planning. In *IEEE/RSJ Int. Conf. Intell. Robot. Sys. (IROS)*, 2020. 1

[28] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. Self-Supervised Learning of State Estimation for Manipulating Deformable Linear Objects. In *IEEE Robot. Autom. Lett.*, volume 5, pages 2372–2379, Apr. 2020. 1

[29] Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, Learning, Perception, and Control Methods for Deformable Object Manipulation. In *Sci. Rob.*, volume 6, pages 1–16, May 2021. 1

[30] Mingrui Yu, Hanzhong Zhong, and Xiang Li. Shape Control of Deformable Linear Objects with Offline and Online Learning of Local Linear Deformation Models. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1337–1343, May 2022. 1

[31] Alan L Yuille and Norberto M Grzywacz. A Mathematical Analysis of the Motion Coherence Theory. *Int. J. Comput. Vis.*, 3(2):155–175, 1989. 1, 3

[32] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847*, 2018. 8