Tracking Branched Deformable Linear Objects Using Particle Filtering on Depth Images

Yuxuan Yang, Johannes A. Stork, and Todor Stoyanov

***Abstract*— Branched deformable linear objects (BDLOs), such as wire harnesses, are important connecting components in manufacturing industries. However, due to deformability, a lack of distinct visual features, and complex branched structure, automating tasks involving these BDLOs remains a challenge. In this paper, we propose a particle-filter-based method to track the state of a BDLO. To circumvent the high cost of tracking the complex high-dimensional BDLO state, we instead track each branch as an individual B-spline. Our method learns a data-driven model to predict the likelihood of each particle conditioned on depth image observation. In contrast to current state-of-the-art approaches based on non-rigid registration, we do not require pre-segmenting the BDLO, thus alleviating a strong and limiting assumption. We train our approach on domain-randomized depth data from simulation and achieve zero-shot transfer to real-world BDLOs, achieving state-of-the-art tracking performance when the pre-segmentation fails.**

I. Introduction

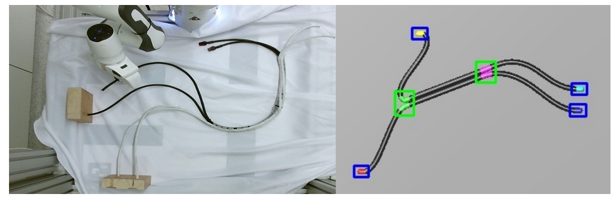
Branched deformable linear objects (BDLOs), as shown in Fig. 1a, are widely used in various manufacturing industries, like aviation or automotive production. They consist of multiple interconnected branches and provide a flexible means of connecting multiple components. However, despite the high degree of automation in these industries, the manipulation of BDLOs has been mostly performed manually, remaining labor-intensive and costly tasks.

State tracking is the foremost of several significant challenges to automating BDLO manipulation tasks. Even tracking a single Deformable Linear Object (DLO) poses difficulties due to its high-dimensional state space, deformability, and limited feature details, as highlighted by Zhu et al. [1]. Complex branched structures of BDLOs further amplify these difficulties.

Despite the advances in tracking a single DLO [2], [3], [4], [5], two notable limitations hinder those methods from being applied in real-world applications. First, current methods mainly focus on tracking individual DLOs, whereas branched deformable linear objects (BDLOs) are less studied but more frequently encountered in industrial settings [6]. Applying these methods to track BDLOs requires additional efforts like modifications [7] or pre-processing [8]. Second, those methods often assume the point sets of a DLO are easily segmented by colors, a condition rarely met in a real-world scenario. Although employing semantic segmentation based on convolutional neural networks trained on synthetic RGB images [9], [10], [11], [12] offers a more general solution for generating DLO masks, it still suffers from a sim-to-real gap and has suboptimal performance when lacking a good color contrast.

This work was partially supported by Vinnova / SIP-STRIM projects 2020-04467 and 2021-04693, and was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Authors are with the Center for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden. [yuxuan.yang, johannesandreas.stork, todor.stoyanov]@oru.se.



(a) (b)

Fig. 1: Wire harnesses. (a) a robot manipulate a wire harness, (b) a simplified example in simulation, the blue boxes labels the endpoints while the green boxes labels the branch points.

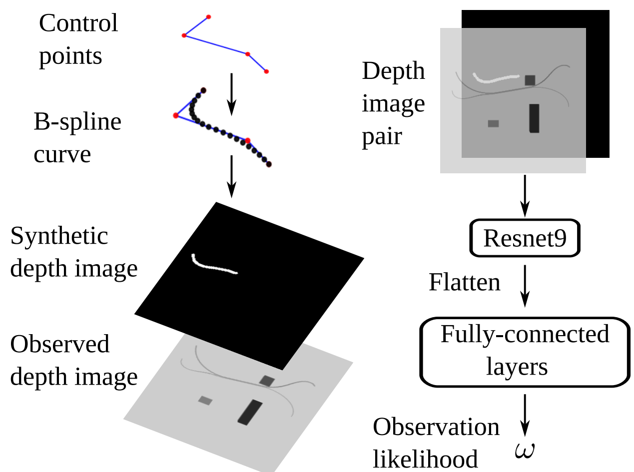
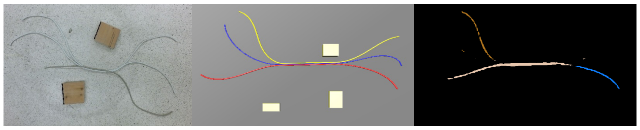


Fig. 2: The overview of the observation likelihood prediction. The goal is to evaluate how likely it is that a set of control points C correspond to a branch in the observed depth image. The likelihood predictor consists of Resnet9 and fully-connected layers. We first generate a B-spline curve based on the control points and project it into a synthetic depth image. Then, we pair the synthetic image with an observed depth image as the input to the predictor.

To address the aforementioned limitations, we propose a method that uses particle filters to track BDLOs. The method consists of two modules. The first module detects BDLOs' feature points like endpoints and branch (intersection) points, as shown in Fig. 1b. This branch feature point detector employs off-the-shelf convolutional neural networks [13]. To aid sim-to-real transfer, the detector utilizes depth images instead of RGB images that are used in [14], [15], [16]. The second module tracks BDLOs by individually tracking each branch, as illustrated in Fig. 2. We approximate each branch as a B-spline represented by a few control points. This makes particle filters for state tracking feasible by considering only



(a) (b) (c)

Fig. 3: Examples of failing to detect DLOs using a learning-based instance segmentation method, Ariadne+ [10]: (a) is an RGB image of a real-world BDLO which Ariadne+ cannot segment mainly because of the low contrast between the background and the BDLO; (b) is an RGB image of BDLO in simulation while (c) is the corresponding mask for BDLO generated by Ariadne+, which fails to correctly capture the branching structure.

the lower-dimensional states of the curves' control point positions rather than the high-dimensional states of the entire BDLO with complex configurations and shapes. We generate a curve from a list of control points sampled by the particle filter, and then project the curve to a synthetic depth image. We devise a data-driven predictor for approximating the likelihood of each particle given the input of a synthetic depth image and the corresponding observed depth image. The main contribution of this paper is thus a method that allows the tracking of BDLOs using particle filtering on depth images without an assumption of having access to a segmented BDLO point set.

II. Related Work

Some recent works [2], [3], [4], [5] have demonstrated systems capable of successfully tracking individual DLOs. However, the assumptions and limitations inherent in these methods hinder their implementation in tracking BDLOs.

A. Tracking by Particle filters

Particle filters are widely used in rigid-object-tracking applications because they can leverage different observation models and dynamics priors [17]. DLOs are typically characterized by a high-dimensional state space that requires an exponentially high number of particles for tracking. "The curse of dimensionality" in particle filters causes computational inefficiency and a high memory requirement, even though the particle filter is highly parallelizable. Lagneau et al. [18] implement a particle filter to track a few control points of a B-spline which approximates the target DLO. In our previous work [5], we learn a low-dimensional latent state space through an autoencoder for individual DLO state estimation and tracking. However, branch configurations of BDLOs may vary significantly. To adequately represent the state of BDLOs, it is necessary to employ a higher-dimensional latent state space. Consequently, the high dimensional state leads to computational scalability issues of particle filters. To avoid this issue, we track each branch separately.

B. Mask-based DLO Tracking

DLO tracking is often formulated as a non-rigid registration problem. One major limitation of current work is the requirement that segmentation masks are available to distinguish between points of a DLO and those from background entities. A color-based segmentation is a common solution based on the assumption of knowing the colors of the DLO and the background [2], [3], [4], [5]. Recently, based on the availability of a public DLO dataset [19], learning-based instance segmentation methods [10], [11] have become popular for the semantic segmentation of DLOs. However, these methods unavoidably suffer from sim-to-real gaps and are not able to provide a robust and complete solution when DLOs intersect or merge into a single DLO (as shown in Fig. 3) [10]. The semantic segmentation in these methods may need to be re-trained or fine-tuned for different DLOs and backgrounds [11].

Instead of requiring segmentation masks for DLOs, we explore the possibility of tracking BDLOs using depth images. Depth images have multiple applications in perception, including 3D object segmentation [20], [21], primitive shape recognition for object grasping [22], [23], and manipulation policy learning [24]. These works suggest that depth images can encode the relevant geometric structures and naturally facilitate sim-to-real transfer [24].

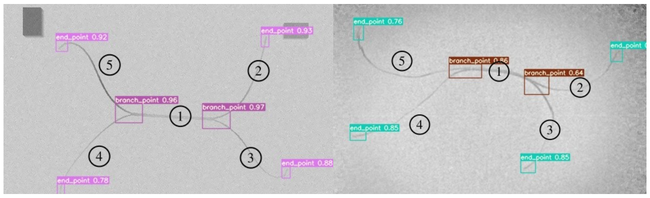
C. Tracking Branched Deformable Linear Objects

Tracking BDLOs is a key element for robot manipulation applications like cable routing and wire harness installation. There are attempts in picking entangled wire harnesses [25], automated optical inspection of wiring harnesses using synthetic data [6], and untangling DLOs [26]. However, regarding tracking, as far as we know, only a few studies directly tackle the BDLO tracking problem.

Wnuk et al. [7] modify the method of structure preserved registration [27] to enhance its capability for tracking BDLOs under occlusion. Their method assumes segmented BDLO point sets and relies on physics simulation where BDLO model parameters, such as Young's modulus and Poisson's ratios, can be difficult to tune. Zürn et al. [28] use self-organizing maps to register a model of the BDLO to a corresponding point cloud, but the computational complexity of the method poses a challenge. Zürn et al. [8] propose to address this by estimating correspondence between a predefined topology of BDLOs and an image acquired by a 3D stereo camera. However, outliers from the image segmentation, as well as errors during skeletonization, may cause mismatches. In contrast, we track BDLOs using particle filters without relying on segmented point sets.

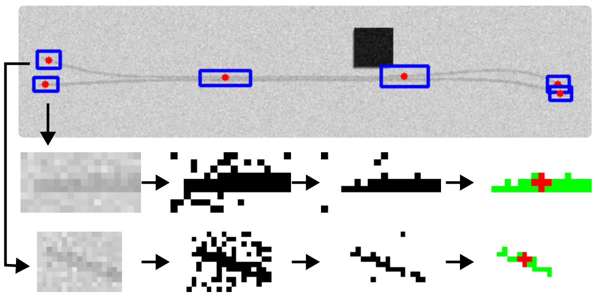
III. Method

In this section, we describe our BDLO state tracking method based on Sequential Importance Resampling (SIR) Particle Filter with a data-driven observation likelihood predictor. First, we introduce a feature points detector for localizing endpoints and branch points (Section III-A); Second, we briefly explain B-splines and why we choose them for state representation (Section III-B); Third, we formulate the tracking problem in a particle filtering framework (Section III-C); Then, we introduce three different motion models for propagating particles in the particle filtering (Section III-C.1); Last, we describe how the observation likelihood predictor works (Section III-C.2).



(a) Synthetic depth image (b) Real-world depth image

Fig. 4: Feature points and branches of branched deformable linear objects. Both images show a BDLO consisting of 5 branches with the feature points labeled by rectangles and branches labeled by numbers.



(a) (b) (c) (d)

Fig. 5: The process of feature point determination. (a) Region of interest in a bounding box; (b) binarization; (c) dilation; (d) locating the centroid of the largest connected component.

A. Feature Points Detector

We introduce two kinds of feature points—endpoints and branch points—to describe the structure of BDLOs. We define a branch in BDLOs as a DLO segment between an endpoint and a neighboring branch point, or between two neighboring branch points, as shown in Fig. 4. Localizing feature points reduces the state dimensions as the points are the control points (Section III-B). Some motion models need feature points to calculate control inputs (Section III-C.1).

Similar to [14], [15], [16], we deploy a Convolutional Neural Network to detect the feature points. In particular, we fine-tuned the YOLOv7 model [13] pre-trained on ImageNet using the synthetic depth images of BDLOs, as shown in Fig 4a, with several data augmentations. The feature point detection module can also employ any external algorithm which is capable of producing detections on the targets. The position of a detected BDLO feature is determined based on the bounding boxes, as illustrated in Fig. 5.

B. B-spline Representation

While an individual DLO has been represented by a B-spline curve in tracking applications [18], it often has a more complex shape to represent when it forms a BDLO with other DLOs as it has to obey continuity constraints at branch points. Consequently, fitting a B-spline in this scenario requires a larger number of control points. To facilitate efficient and effective tracking using particle filtering, we describe each branch of the BDLO as a B-spline curve. This choice leads to a manageable number of control points for each branch, ensuring computational feasibility while accurately capturing the branch's shape during tracking.

A branch in a BDLO is discretized into N nodes, $\boldsymbol{X}_l^t = [\boldsymbol{x}_{l1}^t, \boldsymbol{x}_{l2}^t, ..., \boldsymbol{x}_{lN}^t]$, where $\boldsymbol{x}_{ln}^t \in \mathbb{R}^3$ is the position of the nth node on lth branch at the time step t. The shape of a branch is represented by a B-spline geometric model [18], which is known to efficiently represent complex linear 3D shapes. A clamped B-spline can be constructed from a list of control points, $\boldsymbol{C}_l^t = [\boldsymbol{c}_{l1}^t, \boldsymbol{c}_{l2}^t, ..., \boldsymbol{c}_{lM}^t]$, where the first and the last control points of the curve coincide with the endpoints of the curve, i.e. $\boldsymbol{c}_{l1}^t \equiv \boldsymbol{x}_{l1}^t$, $\boldsymbol{c}_{lM}^t \equiv \boldsymbol{x}_{lN}^t$. Normally $M \ll N$, therefore it describes a DLO in a low-dimensional space. Compared to the latent state space we used in our previous work [5], this B-spline representation is more flexible, since the data-driven method can suffer from poor generalization among DLOs with different lengths and deformation characteristics. In our implementations, we fit a B-spline to a branch and sample points from the curve as $\tilde{\boldsymbol{X}}_l^t = [\tilde{\boldsymbol{x}}_{l1}^t, \tilde{\boldsymbol{x}}_{l2}^t, ..., \tilde{\boldsymbol{x}}_{lN}^t]$. Our goal in tracking is to find the best list of $\boldsymbol{C}_l^{t*}$:

$$\boldsymbol{C}_l^{t*} = \underset{\boldsymbol{C}_l^t}{\arg\min} \, \|\tilde{\boldsymbol{X}}_l^t - \hat{\boldsymbol{X}}_l^t\|_2, \hat{\boldsymbol{X}}_l^t = f_{\text{b-spline}}(\boldsymbol{C}_l^t) \quad (1)$$

where $f_{\text{b-spline}}$ is the function that samples $\hat{\boldsymbol{X}}_l^t$ on a B-spline curve of proposed control points $\boldsymbol{C}_l^t$.

C. Particle Filtering

A particle filter employs a group of particles to describe the posterior distribution of a stochastic process given noisy and/or partial observations. At each time step, particles evolve according to a motion model, or transition probability density $p(\boldsymbol{z}_l^t | \boldsymbol{z}_l^{t-1}, \boldsymbol{u}^t)$, where $\boldsymbol{z}_l^t$ is a particle representing a proposed state and $\boldsymbol{u}^t$ is the control input. Each particle is assigned a weight, ω_l, based on the observation likelihood—the likelihood of each proposed state given the current observation. The particles are subsequently resampled based on the importance weights, which preserves particles with higher weights for the next time step. A weighted sum of all particles approximates the filtering probability density.

Implementing a particle filter directly for BDLO state tracking requires estimating a posterior distribution of each branch state given the current observation. When tracking a BDLO, one of the challenges encountered is the lack of a low-dimensional state representation due to their intricate shapes. To address this issue, we decouple tracking the BDLO into tracking each branch, using L particle filters to track L branches separately. Empirically, we use four control points to describe a branch. Therefore, the branch state is a list of the four control points, $\boldsymbol{C}_l^t$ and the BDLO state is the collection of all L branch state, $\boldsymbol{C}_{\text{BDLO}}^t = \{\boldsymbol{C}_0^t, \boldsymbol{C}_1^t, ..., \boldsymbol{C}_l^t\}$.

In our previous work [5], we approximate the observation likelihood of each particle based on point set registration. However, this approach relies on the availability of segmented DLOs' point sets. In this paper, we train a data-driven predictor instead, which predicts observation likelihoods of proposed B-spline control points based on the observed depth image, $p(\boldsymbol{C}_l^t | \mathcal{I}^t)$. The overview of the predictor is shown in Fig. 2, and the tracking process is described in Algorithm 1.

We initialize L particle filters for tracking L branches. After that, we first propagate the K particles of each filter

Algorithm 1: Tracking m-th branch state in BDLO by particle filtering

Input : observed depth image $\mathcal{I}^t$, previous particles states $\boldsymbol{C}_{l(1:K)}^{t-1}$

Output: current particles states $\boldsymbol{C}_{l(1:K)}^{t}$, importance weights $\omega_{l(1:K)}^{t}$

Update $\hat{\boldsymbol{C}}_{l(1:K)}^{t}$ # Section III-C.1

for $k \leftarrow 1$ **to** K **do**

  $\omega_{lk}^{t} = f_p(\boldsymbol{C}_{lk}^{t}, \mathcal{I}^t)$; # Section III-C.2

$\omega_{l(1:K)} = \frac{\hat{\omega}_{l(1:K)}}{\sum \hat{\omega}_{l(1:K)}}$

if $N_{\texttt{eff}} < K/2$ **then**

  $\boldsymbol{C}_{l(1:K)}^{t} \leftarrow \texttt{resample}(\boldsymbol{C}_{l(1:K)}^{t}, \omega_{l(1:K)}^{t})$

  # Equation 2 and 3

according to the motion model. Then, we predict each particle's observation likelihood, $p(\boldsymbol{C}_l^t|\mathcal{I}^t)$, and update the weights, ω_{lk}. In the end, we use systematic resampling to avoid the degeneracy problem of the algorithm. This resampling method is favorable due to its computational complexity and good empirical performance. Systematic resampling generates K ordered numbers u_{lk} and uses them to select samples $\boldsymbol{c}_{l(1:K)}^t$ by the multinomial distribution.

$$u_{lk} = \frac{(k-1)+\tilde{u}}{K}, \text{with} \quad \tilde{u} \sim \mathrm{U}[0,1) \tag{2}$$

$$\boldsymbol{C}_{lk}^t = \boldsymbol{C}_{li}^{t-1} \text{ with } i \text{ s.t.} u_{lk} \in [\sum_{s=1}^{i-1}\omega_{ls}, \sum_{s=1}^{i}\omega_{ls}] \tag{3}$$

To avoid particle impoverishment in resampling, we compute the effective sample size to determine when to execute the resampling step. The effective sample size evaluates how well the current particles represent the target posterior according to the formulation:

$$N_{\texttt{eff}} = \frac{1}{\sum_{i=1}^{K}(\omega_{li})^2} \tag{4}$$

The higher the variance of the importance weights, the worse the approximation of the target distribution. We resample each time N_{eff} drops below the threshold of $K/2$, where K is the number of particles.

Next, we introduce the two main components in the particle filtering—the motion model and the observation likelihood predictor.

1) Motion Model: A motion model propagates the distribution of the state from the previous time step $t-1$ to the current time step t given the control input $\boldsymbol{u}$.

$$p(\boldsymbol{C}^t|\boldsymbol{C}^{t-1}, \boldsymbol{u}^t) = \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{5}$$

There are different motion model options. When a BDLO's movement is quasi-static, it is reasonable to assume the control points' velocities remain constant between successive time steps. Thus, a constant velocity model is obtained as:

$$p(\boldsymbol{C}^t|\boldsymbol{C}^{t-1}, \boldsymbol{C}^{t-2}) = \mathrm{N}(\boldsymbol{C}^{t-1} + \alpha\Delta\boldsymbol{C}^t, \boldsymbol{\Sigma}) \tag{6}$$

where Σ is a noise covariance matrix that is treated as a tunable parameter of the motion model and $\Delta\boldsymbol{C} = \boldsymbol{C}^{t-1} - \boldsymbol{C}^{t-2}$. When $\alpha = 0$, the model becomes a random walk motion model, where the state at time step t is given by

$$p(\boldsymbol{C}^t|\boldsymbol{C}^{t-1}) = \mathrm{N}(\boldsymbol{C}^{t-1}, \boldsymbol{\Sigma}) \tag{7}$$

These two motion models are simple and widely used but they may not capture the motion patterns accurately.

A slight movement of an endpoint will produce a tiny change in the DLO shape, hence moving the control points that describe the curve. We introduce a curve motion model given the control inputs $\boldsymbol{u}$ which are the detected positions of feature points, or the two endpoints of each branch. (Similar ideas have been implemented in online residual model learning [29] and deformation model used in DLO shape control [30]):

$$\boldsymbol{C}^t = \boldsymbol{C}^{t-1} + \Delta\boldsymbol{C}^t, \quad \Delta\boldsymbol{C}^t = \boldsymbol{J}^t\boldsymbol{U}^t \tag{8}$$

where $\Delta\boldsymbol{C}^t \in \mathbb{R}^{3(M-2)}$ is the change in control points positions, $\boldsymbol{U}^t \in \mathbb{R}^6$ is the movement of a curve segment's two endpoints, and $\boldsymbol{J}^t \in \mathbb{R}^{3(M-2)\times 6}$ is the local curve model matrix relating the two. For simplicity, we omit time t in the following equations. For the ith element of $\Delta\boldsymbol{C}$, we have:

$$\Delta c_i = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{j}_i \tag{9}$$

where $\boldsymbol{j}_i \in \mathbb{R}^6$ is the ith row of $\boldsymbol{J}$. To estimate $\boldsymbol{j}_i$, we collect l consecutive data of Δc_i and $\boldsymbol{u}^{\mathrm{T}}$ online while the tracked DLO segment is moving:

$$\Delta\boldsymbol{C}_i = \begin{bmatrix} \Delta c_i(t-l) \\ \Delta c_i(t-l+1) \\ \vdots \\ \Delta c_i(t-1) \end{bmatrix} \in \mathbb{R}^l, \boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}^{\mathrm{T}}(t-l) \\ \boldsymbol{u}^{\mathrm{T}}(t-l+1) \\ \vdots \\ \boldsymbol{u}^{\mathrm{T}}(t-1) \end{bmatrix} \in \mathbb{R}^{l\times 6} \tag{10}$$

Therefore, we have:

$$\Delta\boldsymbol{C}_i = \boldsymbol{U}\boldsymbol{j}_i \tag{11}$$

Based on the collected data $\Delta\boldsymbol{C}_i$ and $\boldsymbol{U}$, we solve the regularized least squares problem:

$$\hat{\boldsymbol{j}}_i^{\,*} = \underset{\boldsymbol{j}_i}{\operatorname{argmin}} \|\Delta\boldsymbol{C}_i - \boldsymbol{U}\boldsymbol{j}_i\| + \lambda\|\boldsymbol{j}_i\| \tag{12}$$

Which results in an analytical solution for $\hat{\boldsymbol{j}}_i$:

$$\hat{\boldsymbol{j}}_i^* = (\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} + \lambda\boldsymbol{I})^{-1}\boldsymbol{U}\Delta\boldsymbol{C}_i \tag{13}$$

where λ is a regularization weight, and $\boldsymbol{I}$ is the identity matrix.

Given the quasi-static assumption and knowing the control inputs, the curve motion model can predict the motion of control points more accurately than a constant velocity model, however, the latter is more robust when the assumption cannot be maintained, as discussed in Section IV-B.

2) Observation Likelihood Predictor: $p(\boldsymbol{C}_{lk}^t|\mathcal{I}^t)$ measures how likely it is that the k-th list of B-spline control points $\boldsymbol{C}_{lk}^t$ for the l-th BDLO branch would explain the observed depth image. In practice, we approximate this likelihood by $p(\tilde{\mathcal{I}}_{lk}^t|\mathcal{I}^t)$, where $\tilde{\mathcal{I}}^t$ is a synthetic depth image projected from the evaluated particle. A registration-based approach to approximate the observation likelihood in our previous work [5] relies on the assumption of knowing the point sets of the DLO. We use a data-driven predictor to estimate $p(\tilde{\mathcal{I}}_{lk}^t|\mathcal{I}^t)$ given the observed depth image and the synthetic depth image from a proposed B-spline curve, $\hat{\omega}_{lk} = f_p(\mathcal{I}^t, \tilde{\mathcal{I}}_{lk}^t)$, as shown in Fig. 2.

The predictor is trained by supervised learning. We calculate the ground truth observation likelihood based on the difference between the proposed branch B-spline points, $\hat{\boldsymbol{X}}_{lk}^t$, and the ground truth branch B-spline points, $\tilde{\boldsymbol{X}}^t$, as the following:

$$d_{lk} = \|\hat{\boldsymbol{X}}_{lk}^t - \tilde{\boldsymbol{X}}_l^t\|_2, \quad \tilde{\omega}_{lk} = e^{-d_{lk}} \tag{14}$$

where $\hat{\boldsymbol{X}}_{lk}^t$ is the B-spline points of a given set of control points, $\hat{\boldsymbol{X}}_{lk}^t = f_{\text{b-spline}}(\boldsymbol{C}_{lk}^t)$. We use the mean square error loss function in the supervised learning, i.e.

$$L = \frac{1}{n}\sum_{i}^{n}(\hat{\omega}_{lk} - \tilde{\omega}_{lk})^2 \tag{15}$$

The input of the predictor is a two-channel image constructed by an observed depth image, $\mathcal{I}^t$, and a synthetic depth image, $\tilde{\mathcal{I}}_{lk}^t$, projected from a B-spline points $\hat{\boldsymbol{X}}_{lk}^t$. This image is then processed by ResNet9 for feature extraction. Subsequently, based on the extracted features, fully-connected layers predict the observation likelihood of the given control points. We choose this feature extraction instead of directly concatenating features of the observed depth image and proposed control points to predict the observation likelihood because, in practice, training the predictor is more challenging with the latter option. We hypothesize that regressing the likelihood on the pair of depth images allows for better spatial generalization than regressing on depth image features and a list of coordinates.

IV. Experiments

In this section, we evaluate our method's performance on BDLO state tracking across simulated and real-world environments. First, we examine the DLO feature detection performance on both simulated and real BDLOs. Next, we compare the performance of our curve motion model with the constant velocity model in different scenarios. Then, we assess the tracking performance of our method with different motion models and compare it with a modified method of structure preserved registration [7]. Additionally, we test the generalization of our method for tracking BDLOs with different topologies. Last, we present qualitative results on real BDLOs using our method.

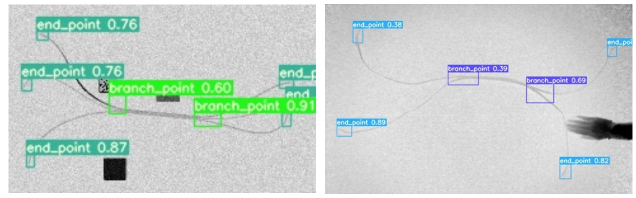


Fig. 6: Feature points detection by YOLOv7 fine-tuned on the synthetic data. left: a depth image from simulation, right: a depth image from real world.

A. BDLO Feature Point Detection

We train the BDLO feature point detector on depth images from simulation. We fine-tune YOLOv7 on depth images by considering them as grayscale images. The training dataset contains 2700 depth images and the validation dataset contains 300 depth images. Each image contains a BDLO consisting of two or three DLOs with different radii and randomly placed obstacles, similar to Fig. 6a. Additionally, the branches in each image vary in length. To mitigate the sim-to-real gap, we generate the depth images in the simulation for training the predictor utilizing the same camera intrinsic parameters as the real one. We also add noise to the depth images for aligning the simulation and real-world depth images more closely.

The trained model is tested on 300 depth images from simulation and 50 depth images from real-world scenarios. The detection results are listed in TABLE I and TABLE II. Examples of boundary boxes detection are shown in Fig. 6.

TABLE I: BDLO feature point detection performance in simulation

Class	Labels	Precision	Recall	mAP_{50}	mAP_{50-95}
All	2100	0.990	0.991	0.995	0.734
Endpoint	1500	0.987	0.982	0.994	0.682
Branch point	600	0.994	1	0.996	0.786

TABLE II: BDLO feature point detection performance in real world

Class	Labels	Precision	Recall	mAP_{50}	mAP_{50-95}
All	275	0.885	0.838	0.795	0.563
Endpoint	250	0.867	0.850	0.742	0.519
Branch point	125	0.90	0.826	0.847	0.608

The position error of endpoint detection is $1.9 \pm 0.5 \times 10^{-2}$m and the error of branch point detection is $2.6 \pm 1.1 \times 10^{-2}$m. One limitation is the uncertainties in BDLO feature point localization, resulting from the gap between the bounding box detection and the subsequent feature point localization process. This gap introduces potential errors and uncertainties in feature points localization and consequently leads to the noises in computing the velocities for the feature points. Thus, it reduces the accuracy of the curve motion model, as discussed in the next section.

B. Curve Motion Model

We investigate the performance of three motion models—curve motion model, constant velocity model, and random walk model—in different scenarios. The results are shown in Fig. 7. In general, the random walk model has the lowest prediction accuracy. In contrast to the latter two models, the curve motion model needs control inputs, i.e. the velocities

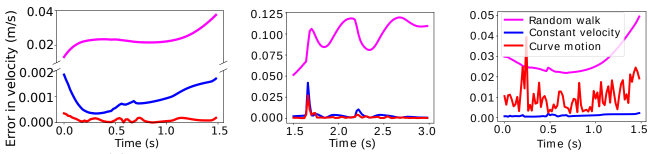


Fig. 7: Comparison of control point velocity predictions using random walk model, constant velocity model, and curve motion model. (a) quasi-static motion given accurate control inputs, (b) non-quasi-static motion given accurate control inputs, (c) non-quasi-static motion given noised control inputs from BDLO feature detection.

of a branch's two ends. First, we assume the control inputs are known, which can occur when a robot manipulates the feature points of a BDLO. The curve motion model exhibits favorable performance when the motion is relatively slow and stable, as shown in Fig. 7a. However, when encountering more dynamic movements, the performance of the curve motion model diminishes, approaching that of the constant velocity model, as shown in Fig. 7b. In situations where the control inputs are unknown, we estimate them by calculating the movement of the BDLO feature points' positions using a feature point detector tested in Section IV-A. The detection introduces noise that subsequently decreases the performance of the curve motion model, as shown in Fig. 7c. The next section will compare the performance of these motion models in particle filtering.

C. Tests on Simulation Data

1) Observation likelihood predictor: The likelihood predictor is based on the structure of the ResNet9 with 3 fully-connected layers subsequently connected. We implement the predictor in PyTorch and use Adam optimizer with a fixed learning rate 1×10^{-4} for training. We collect depth images of BDLOs consisting of 2 DLOs with 2 branch points (similar to Fig. 9(a)). We manipulate the BDLOs by randomly moving the endpoints. We randomly add obstacle objects and randomize DLOs' radii and branches' lengths.

To train the predictor, synthetic depth images are required to construct depth image pairs, as shown in Fig. 2. We get reference control points by fitting a B-spline curve into a branch. Then, we add noises to them. Next, we generate b-spline curves from those noised control points and project the curves into depth images. The loss is calculated as Equation (14), (15).

2) Particle filtering with different motion models: The motion model influences the particle filtering performance. To investigate this, we evaluate the particle filtering in tracking a BDLO using the three different motion models—random walk model, the constant velocity model, and the curve motion model—mentioned in Section III-C.1. Among the motion models considered, the curve motion model requires control inputs, which are the velocities of the BDLOs feature points. The velocities can be computed based on BDLO feature point detection or assumed known when robots are manipulating and moving the feature points.

The experiment results are listed in TABLE III. Increasing the number of particles of the particle filters generally improves the accuracy of the tracking results but it comes

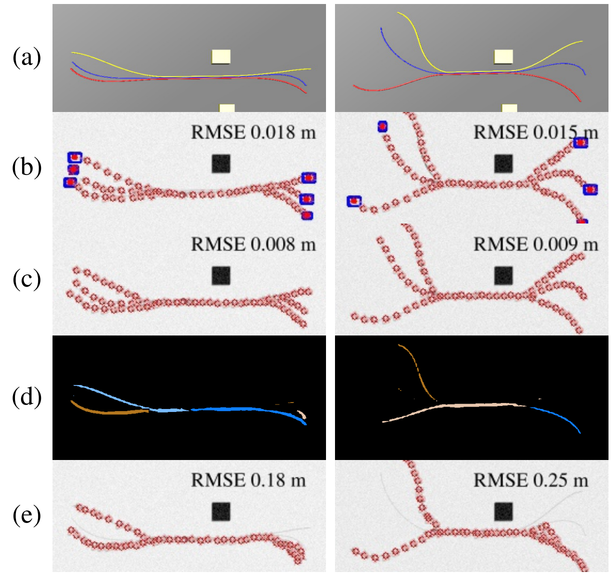


Fig. 8: Comparison of our tracking method to the method based on non-rigid registration. (a) RGB image, (b) our method, (c) non-rigid registration with perfect masks, (d) detected masks, (e) non-rigid registration based on detected masks.

at the cost of increased processing time and memory usage. When accurate control inputs are available, the particle filtering with the curve motion model demonstrates the highest tracking accuracy among the three tested motion models. The model's ability to capture the dynamic behavior of the BDLOs facilitates more accurate tracking. However, the performance of the curve motion model diminishes when the provided control inputs are inaccurate, such as the ones computed based on the feature point detection. In such cases, it only shows a marginal improvement over the random walk model. The experiments indicate that in the absence of accurate control inputs, the constant velocity model, despite its simplicity, yields better tracking results compared to the curve motion model. Therefore, the constant velocity model is employed in the remaining experiments as we do not assume the knowledge of the accurate control inputs. As a trade-off between accuracy and efficiency, we employ 50 particles to achieve a tracking frequency of 10 Hz.

TABLE III: BDLOs tracking performance with different motion models

K	σ	RMSE ($\times 10^{-2}$m) Random walk model	Constant velocity model	Curve motion model [a]	Curve motion model [b]
50	0.2	3.7 ± 2.2	1.2 ± 0.3	1.0 ± 0.3	3.4 ± 2.3
100	0.2	2.8 ± 2.0	1.0 ± 0.2	0.9 ± 0.2	2.6 ± 2.0
200	0.2	2.0 ± 1.6	0.7 ± 0.2	0.7 ± 0.2	1.9 ± 1.8
50	0.5	2.5 ± 1.4	1.6 ± 0.5	1.4 ± 0.5	2.2 ± 1.4
100	0.5	2.0 ± 1.0	1.2 ± 0.4	1.2 ± 0.4	1.9 ± 1.1
200	0.5	1.7 ± 0.7	1.0 ± 0.3	0.9 ± 0.3	1.6 ± 1.0
50	1	4.9 ± 2.6	2.5 ± 1.2	2.3 ± 1.2	4.3 ± 2.2
100	1	3.5 ± 2.0	2.2 ± 1.0	2.1 ± 1.0	3.4 ± 2.0
200	1	3.1 ± 1.7	1.9 ± 0.9	1.8 ± 0.9	2.9 ± 1.6

K: number of particles.
σ: $\Sigma = \sigma \boldsymbol{I}$, where Σ is the covariance matrix and $\boldsymbol{I}$ is an identity matrix.
[a] Assume knowing the ground truth velocities of endpoints and branch points.
[b] Calculate the velocities of endpoints and branch points based on BDLO feature point detection.

3) Comparing with tracking method based on non-rigid registration: Tracking based on non-rigid registration [7]

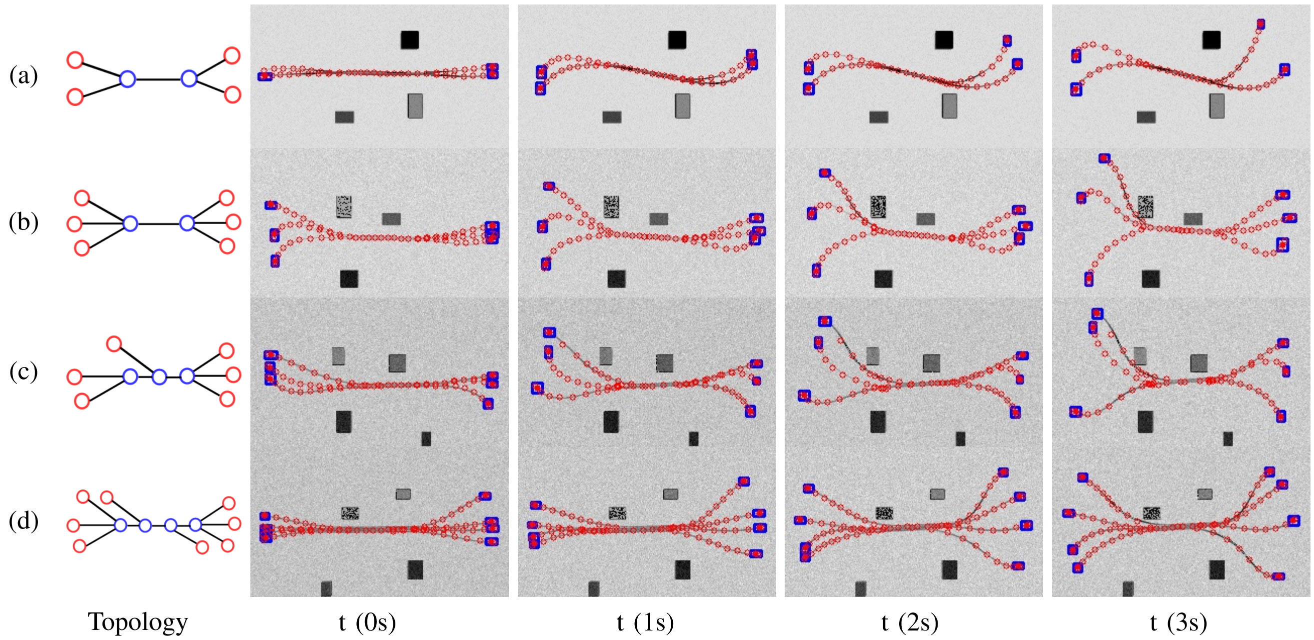


Fig. 9: Qualitative results of tracking performance on BDLOs with different topologies in simulation. (a) 2 DLOs and 2 branch points, (b) 3 DLOs and 2 branch points, (c) 3 DLOs and 3 branch points, (d) 4 DLOs and 4 branch points.

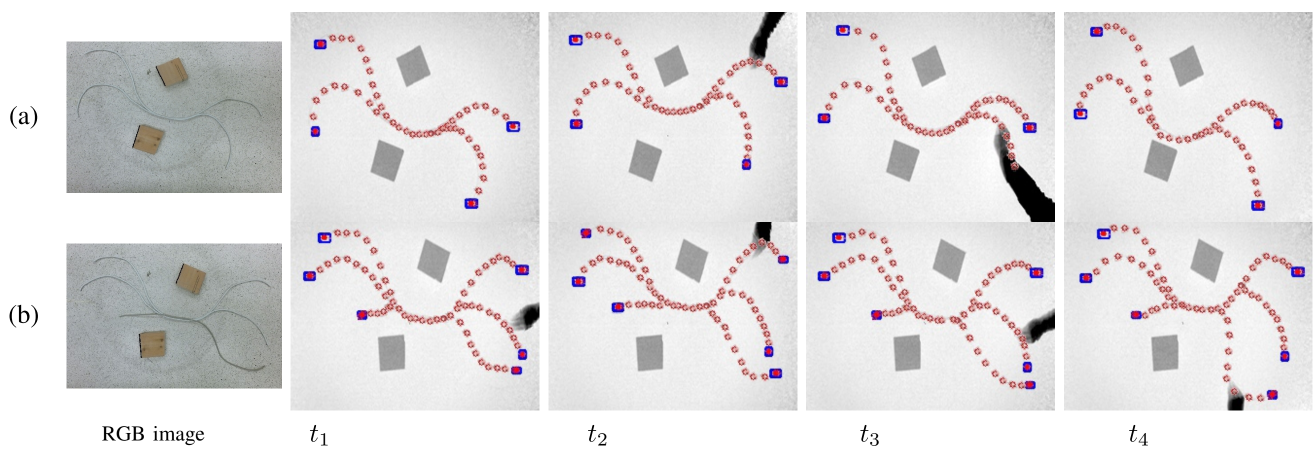


Fig. 10: Qualitative results of tracking performance on BDLOs with different topologies in real-world scenarios. (a) 2 DLOs and 2 branch points, (b) 3 DLOs and 3 branch points.

relies on the assumption of having access to the segmented BDLO point sets. We compare our method with it in two different scenarios, i.e. utilizing perfect masks for BDLO point sets and masks from a segmentation method Ariadne+ [10]. The qualitative comparisons are shown in Fig. 8.

If segmented BDLO point sets are available (Fig. 8(c)), the tracking method based on non-rigid registration performs on par with our tracking method. When a data-driven segmentation method like Ariadne+ generates masks for the BDLO, the masks can be suboptimal (Fig. 8(d)), leading to inaccurate tracking (Fig. 8(e)). The segmentation results can be improved by fine-tuning the segmentation model for a specific scenario, but this requires additional effort in data collection and processing.

4) Adaptability on different topologies: Our method can use the same observation likelihood predictor trained on one topology (Fig. 9(a)) to track BDLOs with different topologies (Fig. 9(b), 9(c), 9(d)). This adaptability arises from the fact that our predictor focuses on individual branches rather than being dependent on specific configurations.

D. Tests on Real-world Data

We conduct real-world experiments to qualitatively evaluate the performance of our method in BDLO tracking, as shown in Fig. 10. In this scenario, pre-segmenting the BDLO is difficult because color-based segmentation or data-driven segmentation does not work. The depth images are captured using a Microsoft Kinect v1 RGB-D camera. We directly employ the observation likelihood predictor trained on the simulation data, as discussed in Section IV-C, without fine-tuning it on the real-world depth images. The performance degrades when occlusions occur during BDLO tracking. The presence of occlusions can also hinder the accurate estimation of the BDLO's state or feature point detection (Fig. 10(a) at time t_3), leading to decreased tracking accuracy.

V. DISCUSSION

We present a method for tracking branched deformable linear objects (BDLOs) utilizing depth images via particle filtering. Unlike existing approaches based on non-rigid registration, our method eliminates the need for pre-segmented BDLO point sets. We incorporate a data-driven observation likelihood predictor within a particle filtering framework. By tracking individual branches of a BDLO, our approach maintains a low-dimensional state space that allows for generalization to different topologies.

Our experiments validate the effectiveness of our proposed method in tracking BDLOs, particularly in scenarios where the masks for BDLO segmentation are imperfect or unavailable. Still, our method has certain limitations. One limitation arises from the uncertainties in localizing BDLO feature points due to the gap between the feature point bounding box detection and precise localization. A framework that provides pixel-level instance segmentation would be preferable. Additionally, our method does not explicitly handle occlusions, and depending on severity, this might be an issue. In future work, we will aim to alleviate these limitations and improve the practical applicability of our method in industrial applications such as wire harness mounting and assembly.

REFERENCES

[1] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li *et al.*, "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 67–77, 2022.

[2] S. Jin, C. Wang, and M. Tomizuka, "Robust deformation model approximation for robotic cable manipulation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 6586–6593.

[3] Y. Wang, D. McConachie, and D. Berenson, "Tracking partially-occluded deformable objects while enforcing geometric constraints," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2021, pp. 14 199–14 205.

[4] C. Chi and D. Berenson, "Occlusion-robust deformable object tracking without physics simulation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 6443–6450.

[5] Y. Yang, J. A. Stork, and T. Stoyanov, "Particle filters in latent space for robust deformable linear object tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 577–12 584, 2022.

[6] H. G. Nguyen, R. Habiboglu, and J. Franke, "Enabling deep learning using synthetic data: A case study for the automotive wiring harness manufacturing," *Procedia CIRP*, vol. 107, pp. 1263–1268, 2022.

[7] M. Wnuk, C. Hinze, M. Zürn, Q. Pan, A. Lechler, and A. Verl, "Tracking branched deformable linear objects with structure preserved registration by branch-wise probability modification," in *2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE, 2021, pp. 101–108.

[8] M. Zürn, M. Wnuk, A. Lechler, and A. Verl, "Tracking deformable objects with point clouds," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2023, pp. 7097–7103.

[9] S. Jin, W. Lian, C. Wang, M. Tomizuka, and S. Schaal, "Robotic cable routing with spatial representation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5687–5694, 2022.

[10] A. Caporali, R. Zanella, D. De Gregorio, and G. Palli, "Ariadne+: Deep learning-based augmented framework for the instance segmentation of wires," *IEEE Transactions on Industrial Informatics*, 2022.

[11] 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.

[12] A. Caporali, K. Galassi, B. L. Žagar, R. Zanella, G. Palli, and A. C. Knoll, "Rt-dlo: Real-time deformable linear objects instance segmentation," *IEEE Transactions on Industrial Informatics*, 2023.

[13] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 7464–7475.

[14] D. D. Gregorio, G. Palli, and L. D. Stefano, "Let's take a walk on superpixels graphs: Deformable linear objects segmentation and model estimation," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 662–677.

[15] M. Zürn, A. Kienzlen, L. Klingel, A. Lechler, A. Verl, S. Ren, and W. Xu, "Deep learning-based instance segmentation for feature extraction of branched deformable linear objects for robotic manipulation," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023, pp. 1–6.

[16] H. Wang and B. Johansson, "Deep learning-based connector detection for robotized assembly of automotive wire harnesses," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023, pp. 1–8.

[17] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1328–1342, 2021.

[18] R. Lagneau, A. Krupa, and M. Marchal, "Automatic shape control of deformable wires based on model-free visual servoing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5252–5259, 2020.

[19] R. Zanella, A. Caporali, K. Tadaka, D. De Gregorio, and G. Palli, "Auto-generated wires dataset for semantic segmentation with domain-independence," in *International Conference on Computer, Control and Robotics (ICCCR)*. IEEE, 2021, pp. 292–298.

[20] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "Unseen object instance segmentation for robotic environments," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1343–1359, 2021.

[21] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg, "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2019, pp. 7283–7290.

[22] P. Schmidt, N. Vahrenkamp, M. Wächter, and T. Asfour, "Grasping of unknown objects using deep convolutional neural networks based on depth images," in *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 6831–6838.

[23] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela, "Using synthetic data and deep networks to recognize primitive shapes for object grasping," in *Proc. of the IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 10 494–10 501.

[24] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg, "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2020, pp. 9411–9418.

[25] X. Zhang, Y. Domae, W. Wan, and K. Harada, "Learning efficient policies for picking entangled wire harnesses: An approach to industrial bin picking," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 73–80, 2022.

[26] X. Huang, D. Chen, Y. Guo, X. Jiang, and Y. Liu, "Untangling multiple deformable linear objects in unknown quantities with complex backgrounds," *IEEE Transactions on Automation Science and Engineering*, 2023.

[27] T. Tang and M. Tomizuka, "Track deformable objects from point clouds with structure preserved registration," *The International Journal of Robotics Research*, p. 0278364919841431, 2018.

[28] M. Zuern, M. Wnuk, A. Schneider, A. Lechler, and A. Verl, "Localization and tracking of deformable linear objects with self organizing maps," in *ISR Europe 2022; 54th International Symposium on Robotics*. VDE, 2022, pp. 1–9.

[29] C. Wang, Y. Zhang, X. Zhang, Z. Wu, X. Zhu, S. Jin, T. Tang, and M. Tomizuka, "Offline-online learning of deformation model for cable manipulation with graph neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5544–5551, 2022.

[30] M. Yu, K. Lv, H. Zhong, S. Song, and X. Li, "Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach," *IEEE Transactions on Robotics*, 2022.