# A Framework for Manipulating Deformable Linear Objects by Coherent Point Drift

Te Tang, Changhao Wang, and Masayoshi Tomizuka

***Abstract*—Manipulation of deformable linear objects is a challenging task for robots. These objects have infinite-dimensional configuration space and are computational-expensive to model, making it difficult for real-time tracking, planning and control. To deal with these challenges, a uniform framework that includes state estimation, task planning, and trajectory planning is proposed in this letter based on the concept of coherent point drift (CPD). A real-time observer is proposed to estimate the states of deformable objects from the perceived point clouds. An online task planner is then developed to recognize the manipulation step according to the state estimation result. For trajectory planning, human operators first train robots example trajectories given several object states. In the test stage, a new feasible trajectory can be autonomously generated by a smooth transformation from training scenarios to test scenarios. A series of rope manipulation experiments on a dual-arm robotic platform are performed to validate the effectiveness of the proposed methods.**

***Index Terms*—Dual arm manipulation, perception for grasping and manipulation, learning from demonstration.**

## I. Introduction

WHILE a great amount of work is focused on the manipulation of rigid objects, manipulating deformable objects, especially deformable linear objects (DLO), remains under-explored. There are many applications involving the manipulation of DLO, such as cable harnessing in factories, thread packing in production lines, suturing in medical surgeries, etc. These tasks are usually labor intensive and have not been automated for many years. The major difficulty lies in the fact that these objects have high degrees of freedom which are expensive to model and control.

Take the rope knotting task as an example (Fig. 1). The objective is to manipulate the rope from a random initial state to a desired knotted state. Robots need to generate corresponding motions to manipulate the rope based on the observation of current rope states. This task has many challenges in several aspects, especially in state estimation, task planning and trajectory planning.

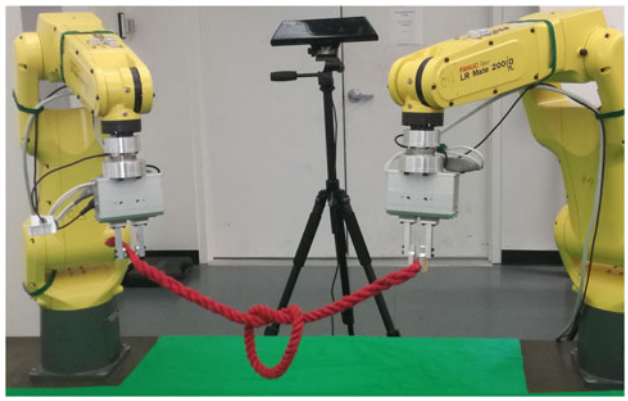

Fig. 1. Two robots knotted a soft rope with real-time visual feedback.

First, for state estimation, the position of each rope segment needs to be identified from 3D camera measurements (point clouds). Usually the rope we are tracking is featureless. In other words, there are no distinguishable markers or features to recognize each segment, and it is unknown which segment on the rope generates the measured points in the point clouds. This missing correspondence makes traditional visual tracking algorithms, for instance Kalman filter, unable to execute. Besides, since the rope is occluded by robot arms or self-occluded by itself frequently during manipulation, the state estimator should be specially designed to handle occlusion robustly. Moreover, considering the curse of dimensionality, running high dimensional state estimations in real time is also a challenging problem.

Second, regarding task planning, robots need to take several sequential steps to knot the rope gradually. Based on the state estimation result, a task planner needs to be developed to classify at which step the manipulation is and determine what following actions each robot should take. Meanwhile, failure detection and recovering mechanism should be included in the task planner in case a failure occurs.

Third, for trajectory planning, the difficulty lies in that the system is underactuated. Limited numbers of grippers (two in our case) are actuating the rope with high degrees of freedom. It is also observed that the rope always runs to unrepetitive shapes during manipulation, i.e., shape differences always exist between training and test scenarios. Therefore, simply replaying the predefined trajectory for training easily fails for the test stages. An online trajectory planner should be developed to refine the trajectory with high efficiency.

Manuscript received February 24, 2018; accepted June 8, 2018. Date of publication July 4, 2018; date of current version August 2, 2018. This work was supported by FANUC Corporation, Japan. This letter was recommended for publication by Associate Editor Y. Pen and Editor J. Wen upon evaluation of reviewers' comments. *(Te Tang and Changhao Wang contributed equally to this work.) (Corresponding author: Te Tang.)*

The authors are with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: tetang@berkeley.edu; changhaowang@berkeley.edu; tomizuka@berkeley.edu).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The Supplementary Materials contain a video showing how the proposed framework enables robots to track and manipulate deformable linear objects robustly. This material is 17.1 MB in size.

Digital Object Identifier 10.1109/LRA.2018.2852770

In this letter, a uniform framework for manipulating deformable linear objects is proposed, which aims at addressing all the challenges discussed above. The core technique we are using is called coherent point drift (CPD) [1], a registration method for mapping one point set to another one non-rigidly. For state estimation, the position of each node on the object is acquired by registering the previous estimation results to the new point cloud measurements. The object states can be estimated robustly in real time under noise, outliers and occlusions. For task planning, CPD is introduced to check the similarity between current object states and pre-recorded training states, then the manipulation step can be determined by finding the maximum similarity. Operation failure can also be detected if the similarity value is below some threshold. For trajectory planning, the learning from demonstration approach [2] is introduced in this letter. In training scenarios, human operators pre-program the corresponding trajectories for some specific rope shapes. During test, a mapping function from the training scenario to the test scenario is constructed by CPD. The training trajectory is warped by the mapping function to obtain a new trajectory which is feasible for the test scenario.

The remainder of this letter is organized as follows. Section II introduces related works on manipulating deformable objects. Section III describes the coherent point drift method for point registration. Section IV explains the design of the framework in detail, which includes state estimation, task planning and trajectory planning modules. Section V tests the performance of the proposed framework by a series of experiments. Supplementary videos can be found in [3]. Section VI concludes the paper and proposes future work.

## II. Related Works

Manipulation of deformable objects is gaining more attention recently because of its broad applications. Morita *et al.* [4] developed a 'knot planning from observation' (KPO) system which estimated the states of ropes, especially the overlap orders by knot theory. Moll *et al.* [5] constructed a minimal energy model to predict the movement of ropes and plan manipulation trajectories. Kudoh *et al.* [6] built a multi-finger hand and programmed skill motions by imitating human knotting procedures. They realized three dimensional in air knotting with diverse types of knots. Many of these methods, however, require empirical laws and are developed for a specific task, which is not easy to generalize for other tasks.

To generalize the manipulation skills, Navarro-Alarcon *et al.* [7], [8] developed a model-free method to automatically servo-control the soft object to a desired shape. A deformation Jacobian matrix which relates the motion of the robot end-effector and the deformation of the object was identified by an online adaptive controller. This matrix was then utilized in generating robot motions given shape errors of the object. Schulman *et al.* [9] proposed to teach robots to manipulate deformable objects from human demonstrations. They implemented the thin plate spline - robust point matching (TPS-RPM) algorithm [10] to warp the original trajectory taught by human demonstration to get a new trajectory which was suitable for the test scene. Several follow-up works further improved this demonstration-based' method. Lee *et al.* [11] extended Schulman's approach by jointly optimizing the registration and the trajectory optimization into a single optimization framework such that the resulting trajectory is smoother. Tang *et al.* [12] implemented TPS-RPM in the object's tangent space to guarantee no over-stretching nor over-compression of the object during manipulation.

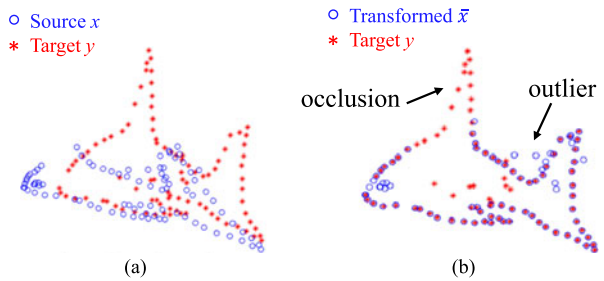


Fig. 2. Illustration of CPD registration. The source point set (blue circle) is registered towards the target point set (red start) by a smooth transformation. (a) Before Registration. (b) After Registration.

For state estimation, a modified expectation maximization (MEM) algorithm was proposed in [13] to track deformable objects from point clouds. They introduced a probabilistic generative model that incorporated observations of the point cloud and the physical properties of the tracked object and its environment. In [14], a simulation database of common deformable garments was proposed to facilitate recognition and manipulation. Mesh models of common deformable garments are simulated with the garments picked up in multiple different poses under gravity, and stored in a database for fast and efficient retrieval.

Most of the above works, however, are trying to deal with one aspect of the challenges (state estimation, task planning and trajectory planning) for manipulating deformable objects. Integrating all these works together to construct a complete framework is another challenging task. The contribution of this letter is that we proposed a uniform framework which addresses all the three major problems with a single technique. The simplicity and consistency of our framework bring great advantages to experimental implementation, parameter tuning, and long-term maintenance.

## III. Non-Rigid Registration by Coherent Point Drift

Non-rigid registration, i.e., aligning one point set to another one non-rigidly, is the core technique we utilize in this letter. There are already several methods developed for non-rigid registration, such as TPS-RPM [10], PR-GLS [15], and CPD [1]. Considering the strong robustness of CPD under occlusion, which is critical in our applications, we choose CPD as our major registration method.

Assume there are two sets of points, source point set $X = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^{N\times D}$ and target point set $Y = \{y_1, y_2, \ldots, y_M\} \in \mathbb{R}^{M\times D}$. $N$ and $M$ are the point numbers in $X$ and $Y$ respectively. $D$ is the point dimension. The objective of CPD is to find a smooth transformation function $v : \mathbb{R}^D \rightarrow \mathbb{R}^D$ to map $X$ to a new position $\bar{X} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N\} \in \mathbb{R}^{N\times D}$ such that $\bar{X}$ is well aligned with $Y$.

Fig. 2 provides an example of CPD registration. The blue source point set is registered towards the red target point set by

a smooth transformation. The point set might contain outliers or miss partial points.

In the above objective, there are two aspects requiring mathematical formulations: (1) how to measure the alignment or similarity between the transformed point set $\bar{X}$ and the target point set $Y$, and (2) how to quantitatively describe the smoothness of the transformation function $v$ from $X$ to $\bar{X}$.

For measuring similarity, a Gaussian mixture model is constructed, where the transformed points in $\bar{X}$ are regarded as the centroids of multiple Gaussians, and points in $Y$ are random samples from the Gaussian mixtures.

Assume that each Gaussian has equal membership probability $\frac{1}{N}$ and consistent isotropic covariance $\sigma^2\mathbf{I}$. The probability of point $y_m$ sampled from the Gaussian mixtures can be calculated by:

$$\begin{aligned} p(y_m) &= \sum_{n=1}^{N} \frac{1}{N} \mathcal{N}(y_m; \bar{x}_n, \sigma^2 \mathbf{I}) \\ &= \sum_{n=1}^{N} \frac{1}{N} \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{\|y_m - \bar{x}_n\|^2}{2\sigma^2}\right) \end{aligned} \tag{1}$$

To account for noise and outliers in the point clouds, an additional uniform distribution is added to the model. The complete mixture model takes the form:

$$p(y_m) = \sum_{n=1}^{N+1} p(n) p(y_m|n) \tag{2}$$

with

$$p(n) = \begin{cases} (1-\mu)\frac{1}{N}, & n = 1, \ldots, N \\ \mu, & n = N+1 \end{cases} \tag{3}$$

$$p(y_m|n) = \begin{cases} \mathcal{N}(y_m; \bar{x}_n, \sigma^2\mathbf{I}), & n = 1, \ldots, N \\ \frac{1}{M}, & n = N+1 \end{cases} \tag{4}$$

where $\mu$ denotes the weight of the uniform distribution.

The complete log-likelihood function $Q$ can be constructed to represent the overall possibility of $Y$ sampling from $\bar{X}$:

$$\begin{aligned} Q &= \sum_{m=1}^{M} \sum_{n=1}^{N+1} p(n|y_m) \log\left(p(n) p(y_m|n)\right) \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N} p(n|y_m) \left(\log\left(\frac{1-\mu}{N(2\pi\sigma^2)^{D/2}}\right) - \frac{\|y_m - \bar{x}_n\|^2}{2\sigma^2}\right) \\ &\quad + \sum_{m=1}^{M} p(N+1|y_m) \log\left(\frac{\mu}{M}\right) \end{aligned} \tag{5}$$

The larger the value of $Q$, the more likely that points in $Y$ are sampled from the Gaussian mixtures created by $\bar{X}$, i.e., the stronger similarity between the two point sets $\bar{X}$ and $Y$.

On the other hand, $\bar{X}$ is transformed from $X$ by a transformation function $v$:

$$\bar{x}_n = x_n + v(x_n) \tag{6}$$

It is desired that $v$ generates globally rigid transformation while also allows locally non-rigid deformation. According to the regularization theory [16] , the smoothness of a function can be measured by the norm $\int_{\mathbb{R}^D} \frac{|V(s)|^2}{G(s)} ds$, where $V(s)$ is the Fourier transform of $v$ and $G(s)$ is a low-pass filter with $G(s) \to 0$ as $s \to \infty$. This Fourier-domain norm basically passes $v$ by a high-pass filter, then measures its remaining power at high frequency. Intuitively, the larger the norm, the more "oscillation" $v$ will behave, i.e., less smoothness.

A modified likelihood function $\tilde{Q}$ is constructed by involving function $v$ and penalizing its oscillation:

$$\begin{aligned} \tilde{Q}(v) &= Q - \frac{\lambda}{2} \int_{\mathbb{R}^D} \frac{|V(s)|^2}{G(s)} ds \\ &= \sum_{m=1}^{M} \sum_{n=1}^{N} p(n|y_m) \log\left(\frac{1-\mu}{N(2\pi\sigma^2)^{D/2}}\right) \\ &\quad - \sum_{m=1}^{M} \sum_{n=1}^{N} p(n|y_m) \frac{\|y_m - x_n - v(x_n)\|^2}{2\sigma^2} \\ &\quad + \sum_{m=1}^{M} p(N+1|y_m) \log\left(\frac{\mu}{M}\right) - \frac{\lambda}{2} \int_{\mathbb{R}^D} \frac{|V(s)|^2}{G(s)} ds \end{aligned} \tag{7}$$

$\lambda \in \mathbb{R}^+$ is a trade-off weight which balances the data fitting accuracy (from $\bar{X}$ to $Y$) and the smoothness requirement (from $X$ to $\bar{X}$). The negative sign before $\lambda$ indicates that a smaller norm, or a smoother transformation from $X$ to $\bar{X}$, is preferred.

It can be proved by variational calculus that the maximizer of (7) has the form of the radial basis function [1]:

$$v(z) = \sum_{n=1}^{N} w_n g(z - x_n) \tag{8}$$

where kernel $g(\cdot)$ is the inverse Fourier transform of $G(s)$, and $w_n \in \mathbb{R}^D$ is unknown kernel weights. In general, kernel $g(\cdot)$ can take any formulations, as long as it is symmetric and positive, and $G(s)$ behaves like a low-pass filter. For simplicity, a Gaussian kernel $g(\cdot)$ is chosen, with $g(z - x_n) = \exp(-\frac{\|z-x_n\|^2}{2\beta^2})$. $\beta \in \mathbb{R}^+$ is a manually tuned parameter which controls the rigidity of function $v$, where large $\beta$ corresponds to rigid transformation, while small $\beta$ produces more local deformation.

We can simplify (7) using the formulation in (8) to produce

$$\begin{aligned} \tilde{Q} &= \sum_{m=1}^{M} \sum_{n=1}^{N} p(n|y_m) \log\left(\frac{1-\mu}{N(2\pi\sigma^2)^{D/2}}\right) \\ &\quad - \sum_{m=1}^{M} \sum_{n=1}^{N} p(n|y_m) \frac{\|y_m - x_n - \sum_{k=1}^{N} w_k g(x_n - x_k)\|^2}{2\sigma^2} \\ &\quad + \sum_{m=1}^{M} p(N+1|y_m) \log\left(\frac{\mu}{M}\right) - \frac{\lambda}{2} trace\left(\mathbf{W}^T \mathbf{G} \mathbf{W}\right) \end{aligned} \tag{9}$$

where $\mathbf{G} \in \mathbb{R}^{N \times N}$ is a symmetric positive Gramian matrix with element $\mathbf{G}_{ij} = g(x_i - x_j)$. $\mathbf{W} = [w_1, \ldots, w_N]^T \in \mathbb{R}^{N \times D}$ is the vectorization of kernel weights in (8).

$\tilde{Q}$ is now parameterized by $(\mathbf{W}, \sigma^2)$ in (9). EM algorithm [17] can be performed to maximize the value of $\tilde{Q}$ and to estimate the parameter $(\mathbf{W}, \sigma^2)$ iteratively. In E-Step, the posteriori probability distribution $p(n|y_m)$ is calculated using the estimated $(\mathbf{W}, \sigma^2)$ from the last M-step. In M-Step, take $\frac{\partial \tilde{Q}}{\partial \mathbf{W}} = 0$ and $\frac{\partial \tilde{Q}}{\partial \sigma^2} = 0$ to achieve a new estimation of $(\mathbf{W}, \sigma^2)$. The closed-form solution for M-step requires some linear algebraic derivation, and more details can be found in [1].

When $\tilde{Q}$ has converged, the transfer function $v$ can be calculated by (8), and the transformed point set $\bar{X}$ can be obtained by matrix multiplication:

$$\bar{X} = X + \mathbf{GW} \tag{10}$$

## IV. Robotic Manipulation of Deformable Linear Objects

This section introduces the framework for robotic manipulation of deformable linear objects. Three major modules, state estimation, task planning and trajectory planning, are introduced in sequence. Each of them uses CPD as a primary tool. For the ease of illustration, an example of rope manipulation will be discussed in the following sections. However, the proposed framework should be general for other types of linear objects as well.

### A. State Estimation of Deformable Linear Objects

During the process of rope manipulation, since the soft rope easily deforms to unscheduled shapes, it is necessary to close the execution loop by monitoring the rope states in real time.

Tracking infinite-dimensional configuration space is impractical. Therefore, we first discretized the rope to a chain of connected nodes with uniform distance (Fig. 7(c)[1]). Our objective is to estimate the position of each node at each time step from the dense, noisy and occluded point clouds (Fig. 7(b)) perceived by stereo cameras.

Suppose at the time step $t$, the rope state is noted as $X^t = \{x_1^t, x_2^t, \ldots, x_N^t\} \in \mathbb{R}^{N\times 3}$, where $x_n^t \in \mathbb{R}^3$ is the $n$th node's position in the three dimensional Cartesian space. $N$ is the total number of nodes. At the next time step $t+1$, the rope is changed to a new state, and its point cloud $Y^{t+1} = \{y_1^{t+1}, y_2^{t+1}, \ldots, y_M^{t+1}\} \in \mathbb{R}^{M\times 3}$ is captured by cameras. $y_m^{t+1} \in \mathbb{R}^3$ denotes the position of a single point in the cloud. $M$ is the total number of points and usually $M \gg N$. By applying CPD registration as dsescribed in Section III, the node positions $X^t$ can be smoothly registered towards the point cloud $Y^{t+1}$, and the transformed point set $\bar{X}$ can be calculated by (10). We use $\bar{X}$ to serve as the state estimation of rope nodes at the time step $t+1$, i.e., $X^{t+1} \triangleq \bar{X}$.

Running the above procedure iteratively, the state estimation at the current time step can always be achieved by registering the previous step estimation towards the current point cloud measurement. Fig. 3 shows the closed-loop structure of this state estimator. Note that since tracking is performed in sequences, and the rope shapes between adjacent time steps should not deviate much, only a few iterations of EM updates will register $X^{t-1}$ to $Y^t$. Therefore, the proposed state estimator can run efficiently in real time. Besides, the estimator is robust to occlusions. During robot manipulation, the view of the stereo cameras might be occluded by the robot arms, which results in missing points in the measured point cloud. However, since the transformation function $v$ is applied on source points coherently, $X^{t-1}$ can still be registered to the missing point area in $Y^t$, i.e., the node position in the occluded area is still able to be obtained (Fig. 7).

To further improve the estimation accuracy and robustness, physics engines can be involved to refine the estimation result by rendering kinematics and dynamics constraints on the virtual rope. More details can be found in our previous work [18].

[1]Fig. 7 is the experimental sanpshot presented in Section V. It is referred here in advance for illustration.

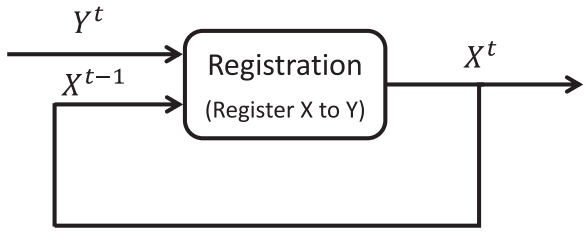


Fig. 3. Framework of point set registration. $Y^t$ is the perceived point cloud at time step $t$. $X^{t-1}$ is the state estimation at previous step. A new estimation $X^t$ is achieved by registering $X^{t-1}$ to $Y^t$.

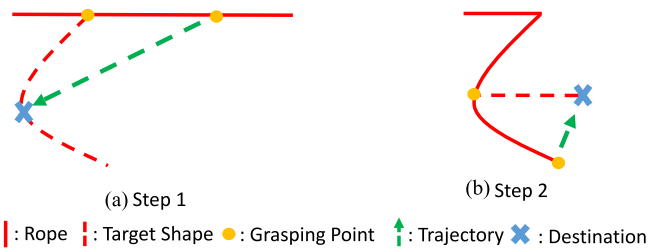


Fig. 4. Two steps to move a straight line to a 'Z' shape. (a) Step 1. (b) Step 2.

### B. Task Planning

A complete manipulation task is usually composed of multiple sequential procedures. For example, as shown in Fig. 4, two major steps are required to move the rope from a straight line to a 'Z' shape. At each step, a corresponding trajectory can be programmed by human operators to guide the robot to successfully manipulate the rope.

For autonomous manipulation, it is necessary for the robot to recognize at which procedure the current state lies in, so that the most relevant trajectory can be selected for the following manipulation.

The CPD registration is utilized again to design this task planner. Suppose that during training, there are $S$ procedures in total to manipulate the rope, and the initial state of the rope at each procedure is recorded as $X_s \in \mathbb{R}^{N\times 3}, s = 1, \cdots, S$. During test, the current state of the rope, $X^t$, is estimated by the proposed observer in Section IV-A. CPD is then applied to

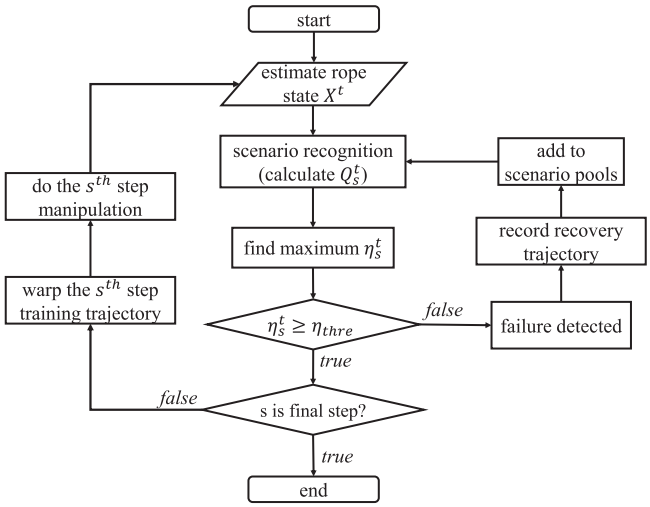


Fig. 5. Framework of task planning.

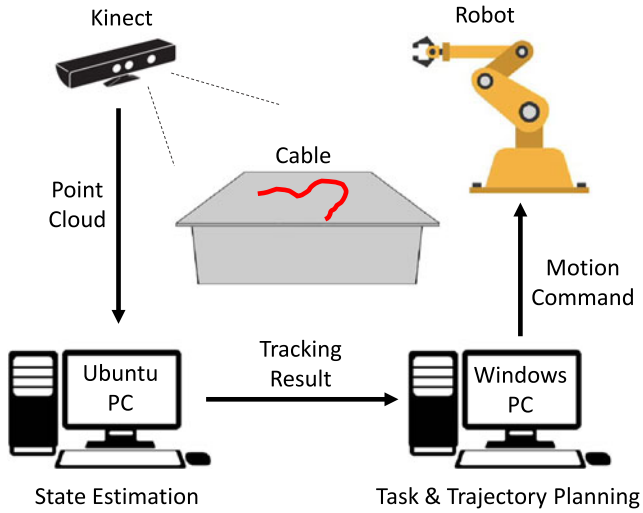


Fig. 6. The testbed setup.

register each recorded state $X_s$ to the current state $X^t$. The log-likelihood function $Q_s^t$ can be calculated after each registration by (9). Note that $Q_s^t$ is negative, and the less negative $Q_s^t$ is, the more similar between $X_s$ and $X^t$. To normalize the similarity within the 0–100% range, a similarity matrix $\eta_s^t$ is defined as follows:

$$\eta_s^t = \frac{Q_t^t}{Q_s^t} \tag{11}$$

where $Q_t^t$ is the log-likelihood calculated by registering $X^t$ towards itself by CPD.

$\eta_s^t$ approaching to 100% indicates stronger similarity between $X_s$ and $X^t$. The most probable step that the current manipulation lies in can be determined by finding maximum similarity:

$$s^* = \arg\max_s \ \eta_s^t, \quad s = 1, \cdots, S \tag{12}$$

Moreover, the task planner can be applied to detect failures during manipulation. If the maximum similarity $\eta_{s^*}^t$ is smaller than a pre-defined threshold, $\eta_{thre}$, it indicates that the current rope state differs from all the scheduled steps. Rope manipulation runs into some unknown failures. The human operator needs to interfere and teach robots recovering trajectories to move the rope back to one of the recorded states. The failure state will also be augmented to the scenario pools $X_s$. If this similar failure occurs again in the future, no human interference is required, instead this failure will be recognized, and the planner will use the taught trajectory for recovering as the last time. The framework of the proposed task planning module is shown in Fig. 5.

### C. Trajectory Planning

During training, for each of the $S$ manipulation procedures, the human operator will program a corresponding trajectory $T_{\text{train}}^s$, $s = 1, \cdots, S$ for the robot end-effector. At test, robots succeed to recognize that the current rope is at the $s$-th step by the task planning module (Section IV-B). However, the $s$-th step's corresponding trajectory $T_{\text{train}}^s$ cannot be directly applied for the test scenario, since no matter how similar, there is always some minor shape difference between the rope at training and that at test. This minor difference makes the exact replay of the training trajectory fail at test: e.g., failing to grasp the rope. Therefore, $T_{\text{train}}^s$ only serves as an approximate reference, while some trajectory refinement based on $T_{\text{train}}^s$ is required to achieve a feasible manipulation at test time.

Note that when registering $X_s$ to the current rope shape $X^t$ during task planning, besides the likelihood function $Q_s^t$, the transformation function $v : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is also constructed by (8). $v$ transforms $X_s$ to align to $X^t$ by twisting the overall Cartesian space. Similarly, the trajectory $T_{\text{train}}^s$ corresponding to $X_s$ can be twisted by $v$ as well to get a new trajectory $T_{\text{test}}^s$ that is feasible for the test scenario.

The trajectory $T$ of a robot end-effector can be regarded as a sequence of poses $\{p, R\}$, where $p \in \mathbb{R}^3$ is the position vector of the end-effector, and $R \in \text{SO}(3)$ is the orientation matrix. With this observation, the feasible trajectory $T_{\text{test}}^s$ can be achieved by applying the following transformation on $T_{\text{train}}^s$:

$$p_{test} = p_{\text{train}} + v(p_{\text{train}}) \tag{13}$$

$$R_{test} = orth(J_v(p_{\text{train}}) \cdot R_{\text{train}}) \tag{14}$$

$J_v(p)$ is the Jacobian matrix of $v$ evaluated at position $p$, and $orth(\cdot)$ is a function that orthogonalizes matrices. If $v$ is a rigid transformation $T_v$, this trajectory transformation procedure is equal to left-multiplying each end-effector pose $T_{\text{train}}$ by $T_v$.

## V. Experiments and Results

A series of experiments were performed to test the proposed state estimation, task planning and trajectory planning algorithms for manipulating soft ropes. Two FANUC LR-Mate 200iD robots collaboratively knotted a 1-meter-long rope from random initial shapes. Experimental videos can be found in [3].

### A. State Estimation

As shown in Fig. 6, the Microsoft Kinect was utilized to monitor the environment at 10 Hz. The captured $640 \times 480$

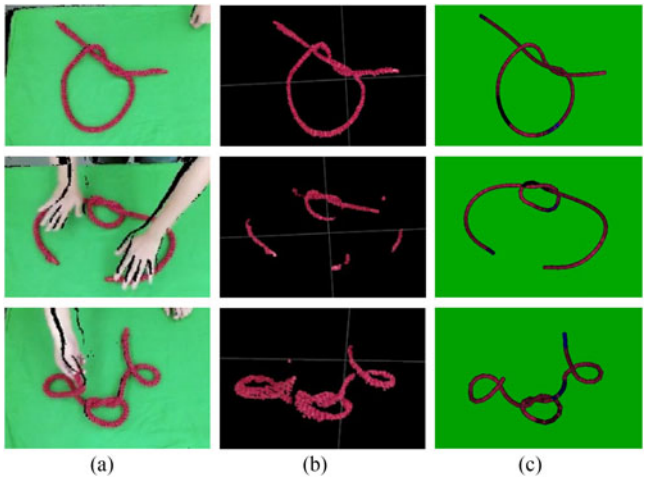


Fig. 7. Snapshots during the real-time tracking experiments. (a) Overall Point Cloud. (b) Segmented Cloud. (c) Tracking Result.

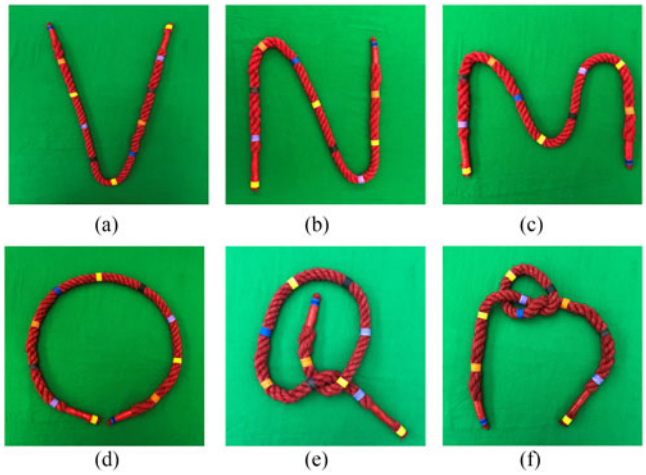


Fig. 8. Six configuration shapes with markers attached. (a) V shape. (b) N shape. (c) M shape. (d) Circle. (e) Half knot. (f) Knot.

TABLE I
EXECUTION TIME OF STATE ESTIMATION

| | |
|---|---|
| Segmentation | 7 ms |
| Downsampling | 2 ms |
| CPD Registration | 10 ms |
| Total | ~20 ms |

RGB and depth images were sent to a Ubuntu 14.04 desktop (Intel i7@3.60 GHz + RAM 16 GB) synchronously and then synthesized to get the environmental point cloud. Since object segmentation was not our focus in this work, we simply placed a red rope on a green or white color background and implemented a color-based filter to segment out the rope from the environment. The rope's point cloud was then downsampled to 200 points uniformly by VoxelGrid filter.

For state estimation, the 1-meter rope was discretized and represented by 50 linked capsules. A CPD toolbox implemented with C++ [19] was utilized to register the 50 nodes' positions towards the rope's point cloud. The point sets were first normalized to zero mean and unit variance before registration. The weight $\mu$ for uniform distribution was chosen to be 0.1. Smoothness regularization parameter $\lambda$ and Gaussian kernel's variance $\beta$ were set as 3.0 and 2.0 respectively. All the data points were denormalized after registration.

Fig. 7 shows the real-time tracking results. Note that the rope's point cloud was noisy, containing outliers and occluded by obstacles from time to time, but the proposed state estimator could still track the rope robustly and efficiently. Table I lists the major execution time of the state estimator. The overall running time is less than 20 ms.

As shown in Fig. 8, to analyze the tracking accuracy, 11 markers with distinct colors were attached on the rope with 10 cm interval. These markers were distinguished by a color-based filter and their ground-truth positions were measured directly from Kinect. Note that these markers were only used for the purpose of ground truth. They were not utilized in our state estimation algorithm. The estimation results from our proposed method

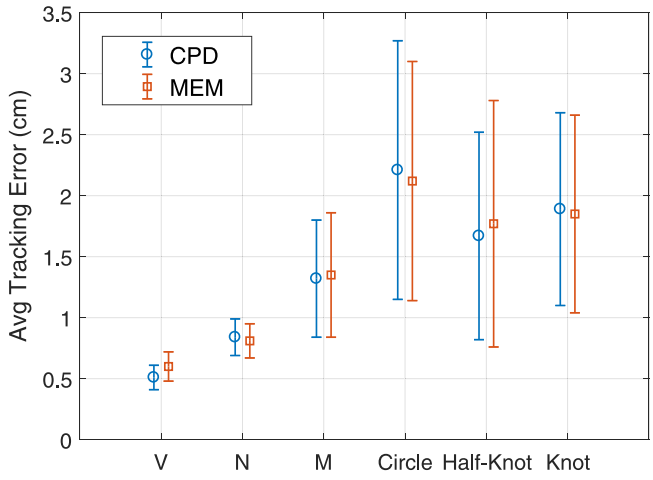


Fig. 9. Average tracking errors and standard deviation at the marker positions.

were compared with these ground-truth values. Fig. 9 shows the average tracking error and standard deviation at six different rope configurations. In general, the tracking error is less than 2.2 cm, smaller than the jaw width (6 cm) of the robot gripper. Therefore, even if the gripper went to an inaccurate grasp pose because of the tracking error, the rope was still located between the gripper's fingers and could be successfully grasped. We also compared our tracking algorithm with the MEM method [13]. Fig. 9 shows that their tracking performance is similar. However, our proposed framework advances on the extendibility since its application is not limited in state estimation, but can also be applied on task planning and trajectory planning for deformable object manipulation.

### B. Task Planning

Following the pipeline in Fig. 6, the state estimation result was then sent to a Windows 10 desktop (Intel i7@3.60 GHz + RAM 8 GB) which ran the task planning and trajectory planning

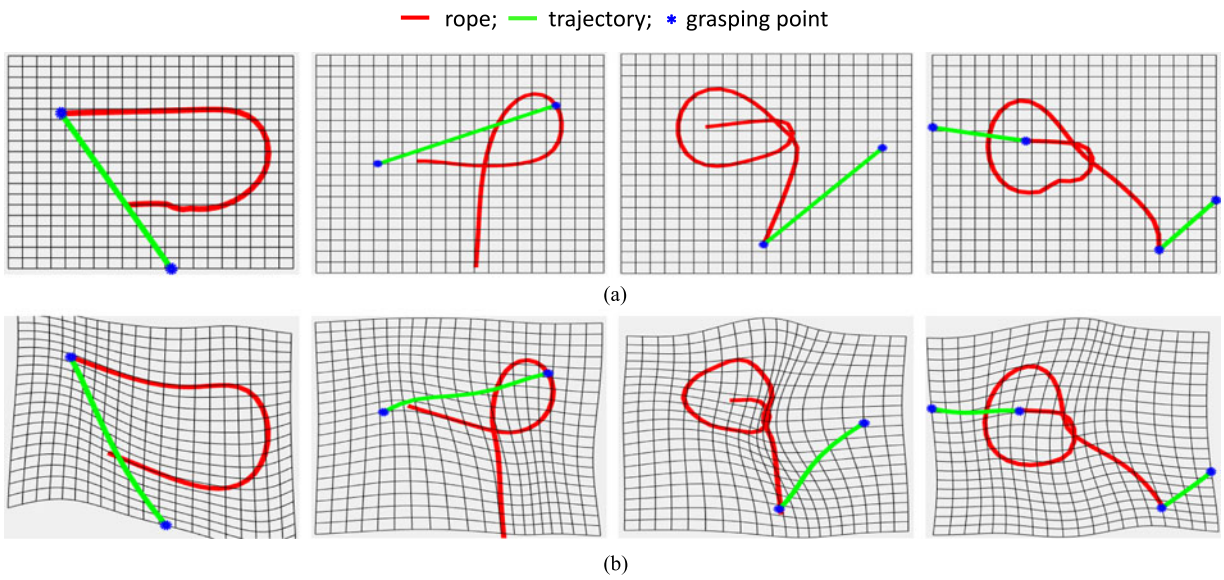


Fig. 10. Four major steps for rope knotting manipulation. Red lines are the rope states, and green lines are the trajectories of the robot end-effector. Blue dots are the grasping/releasing positions. Black grids show that the original Cartesian space is twisted so as to map the training scenarios towards test scenarios. (a) Training Scenarios. (b) Test Scenarios.

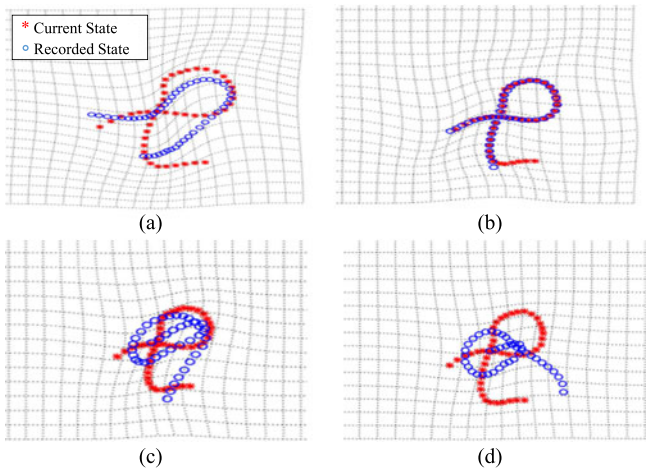


Fig. 11. Similarity check between the current rope state (red dots) and the four recorded training states (blue dots). The second scenario is most similar to the current state with a 90% similarity. (a) Comparison with Step 1, $\eta_1 = 23\%$. (b) Comparison with Step 2, $\eta_2 = 90\%$. (c) Comparison with Step 3, $\eta_3 = 37\%$. (d) Comparison with Step 4, $\eta_4 = 13\%$.

algorithms. ROS [20] served as the interface to communicate between the Kinect, the Ubuntu PC and the Windows PC.

As shown in Fig. 10(a), four major steps were predefined by human operators for the task of rope knotting. At each step, the initial shape of the rope was recorded by the state estimator. The corresponding manipulation trajectory was then demonstrated by lead through teaching. To be specific, operators guide the two robots' end-effectors to go through some waypoints, and the training trajectory was obtained by linear interpolation between neighbour poses.

At test, the current rope states were estimated and compared by CPD to each of the four recorded templates. The similarity level was calculated by (11). As shown in Fig. 11, the red point set (current state) and the blue point set (recorded state) in the second image had the largest similarity (90%), which indicated that the manipulation process was at the second step at that moment. The similarity lower-bond $\eta_{\text{thre}}$ was set as 80%. If all the similarity check is below 80%, a warning message will be shown to ask the human operator to demonstrate a recovering trajectory. The failure states and recovering trajectories were then augmented into the task planning sample pools. The robot's ability of detecting and recovering from failures is shown in the attached videos.

### C. Trajectory Planning

After identifying the task step, the manipulation trajectory was generated by the proposed trajectory planning algorithm.

To represent the rope positions and the manipulation trajectory under the same coordinate system, the relative translation (extrinsic parameter) between the Kinect and the robot world frame is calibrated first. The rope states were all translated to the robot world frame afterwards.

As shown in Fig. 10, the rope state during test was different from that in training. With CPD registration, the training trajectory was transformed by (13) and (14) at each step to achieve the test trajectory. The end-effector poses were then transformed to robot joint command by robotic inverse kinematics. The joint command was finally sent to the robot controller for execution.

Fig. 12 shows the snapshots of autonomous rope knotting by two robot arms. Three types of knotting were designed, with each type tested 15 times. The overall success rate was 40/45. Most of the failure was miss-grasping, which might result from the relatively low accuracy of Kinect and calibration error between the camera frame and robot base frame.

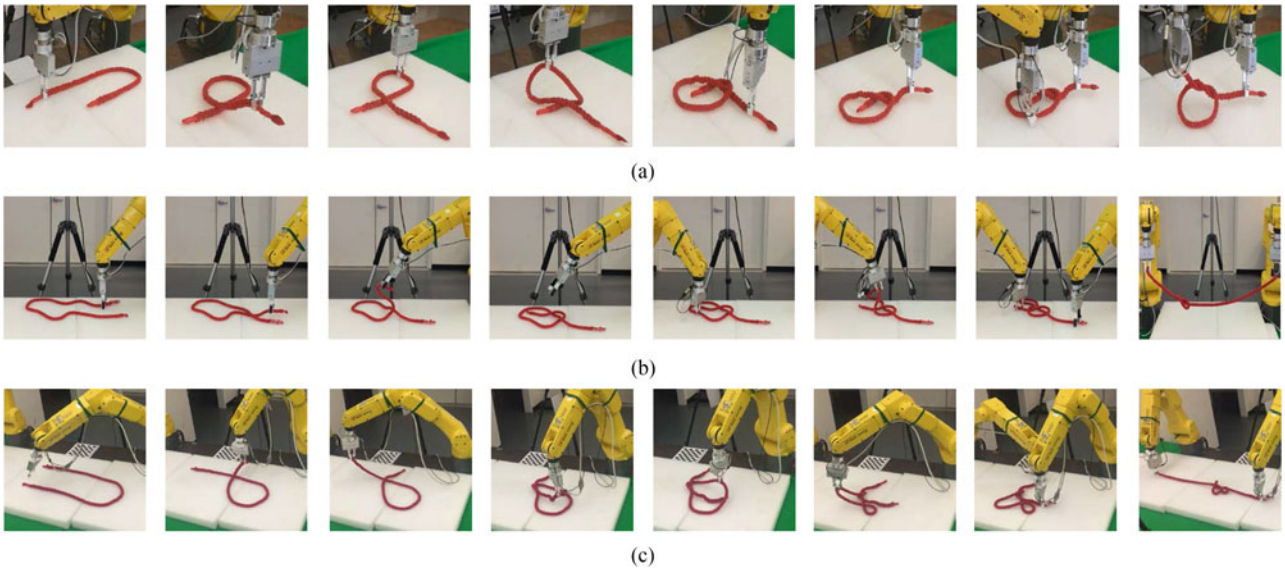


Fig. 12. Snapshots of the rope knotting experiments. Two robot arms were collaborating to knot the rope based on the transformed trajectory. (a)(b)(c) show three different types for knotting. (a) Knot Type 1. (b) Knot Type 2. (c) Knot Type 3.

## VI. Conclusion and Future Work

A uniform framework, which includes state estimation, task planning and trajectory planning, is proposed in this letter for manipulating deformable linear objects. Based on the concept of coherent point drift (CPD), a real-time observer is developed to estimate the node position of the rope by registering the last step estimation towards the current point cloud measurement. A task planner is then developed to let robots recognize at which procedure the current manipulation is by registering training scenarios towards the test scenario. Finally, utilizing the transformation function constructed during scenario registration, the training trajectory can be warped to achieve a new trajectory which is feasible for the test. A series of experiments on ropes knotting tasks are implemented, which indicate the effectiveness of the proposed methods.

The performance of state estimation was compared with other state-of-the-art methods. In the future, we will compare the task planning and trajectory planning modules with other studies as well to make the evaluation more complete.

## References

[1] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[3] Experimental Videos for Manipulating Deformable Linear Objects. 2018. [Online]. Available: http://me.berkeley.edu/%7Etetang/RAL2018/RopeManipulation.html

[4] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Knot planning from observation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2003, vol. 3, pp. 3887–3892.

[5] M. Moll and L. E. Kavraki, "Path planning for deformable linear objects," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 625–636, Aug. 2006.

[6] S. Kudoh, T. Gomi, R. Katano, T. Tomizawa, and T. Suehiro, "In-air knotting of rope by a dual-arm multi-finger robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 6202–6207.

[7] D. Navarro-Alarcon *et al.*, "Automatic 3-d manipulation of soft objects by robotic arms with an adaptive deformation model," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 429–441, Apr. 2016.

[8] D. Navarro-Alarcon, Y.-h. Liu, J. G. Romero, and P. Li, "On the visual deformation servoing of compliant objects: Uncalibrated control methods and experiments," *Int. J. Robot. Res.*, vol. 33, no. 11, pp. 1462–1480, 2014.

[9] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Proc. 16th Int. Symp. Robot. Res.*, Springer, Cham, 2016, pp. 339–354.

[10] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Comput. Vis. Image Understanding*, vol. 89, no. 2, pp. 114–141, 2003.

[11] A. X. Lee, S. H. Huang, D. Hadfield-Menell, E. Tzeng, and P. Abbeel, "Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 4402–4407.

[12] T. Tang, C. Liu, W. Chen, and M. Tomizuka, "Robotic manipulation of deformable objects by tangent space mapping and non-rigid registration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 2689–2696.

[13] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 1130–1137.

[14] Y. Li *et al.*, "Model-driven feedforward prediction for manipulation of deformable objects," *IEEE Trans. Automat. Sci. Eng.*, to be published.

[15] J. Ma, J. Zhao, and A. L. Yuille, "Non-rigid point set registration by preserving global and local structures," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 53–64, Jan. 2016.

[16] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, 1995.

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B (Methodological)*, vol. 39, pp. 1–38, 1977.

[18] T. Tang, Y. Fan, H.-C. Lin, and M. Tomizuka, "State estimation for deformable objects by point registration and dynamic simulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2427–2433.

[19] C++ implementation of the Coherent Point Drift algorithm. 2016. [Online]. Available: https://github.com/gadomski/cpd

[20] The Robot Operating System (ROS). 2009. [Online]. Available: http://www.ros.org/