Model-based Reinforcement Learning Approach for Deformable Linear Object Manipulation

Haifeng Han[1], Gavin Paul[2], and Takamitsu Matsubara[1]

***Abstract*— Deformable Linear Object (DLO) manipulation has wide application in industry and in daily life. Conventionally, it is difficult for a robot to manipulate a DLO to achieve the target configuration due to the absence of the universal model that specifies the DLO regardless of the material and environment. Since the state variable of a DLO can be very high dimensional, identifying such a model may require a huge number of samples. Thus, model-based planning of DLO manipulation would be impractical and unreasonable. In this paper, we explore another approach based on reinforcement learning. To this end, our approach is to apply a sample-efficient model-based reinforcement learning method, so-called PILCO [1], to resolve the high dimensional planning problem of DLO manipulation with a reasonable number of samples. To investigate the effectiveness of our approach, we developed an experimental setup with a dual-arm industrial robot and multiple sensors. Then, we conducted experiments to show that our approach is efficient by performing a DLO manipulation task.**

Fig. 1: Baxter manipulating a rope to achieve the target shape.

I. INTRODUCTION

Deformable Linear Object (DLO)s, such as ropes, cables, wires, etc., are widely used in industry and daily life. Advancement of DLO manipulation by means of robots can benefit many application domains, such as construction, manufacturing, medical surgery, and assisted living [2]. However, it is a challenge for robots to manipulate DLOs which take many different shapes when external forces are exerted upon them. Such diversity of behavior brings difficulties in robot perception and action planning.

Previous work has generally utilized or focused on the model-based planning approach, which assumes a DLO model mainly based on expert knowledge, such as physics [3] or topology [4]. Some work has shown impressive performance with certain settings when conducting compliant tasks, such as knot tying [4] [5]. However, the most significant limitation of model-based planning is that the planning result is highly dependent on the effectiveness and accuracy of the model, which is not only confined by the expert knowledge, but also differs for DLOs' materials and the specific environment. Moreover, the presupposition of model-based planning is a model that specifies all DLO configurations, which is impractical since the state of such a model must necessarily have an intractably high dimensionality.

On the other hand, machine learning has been employed to achieved DLO manipulation by means of a shown demonstration in advance [6]. This machine learning-based approach avoids the need for a high-dimensionality DLO model by focusing on learning by registering demonstrations into a new situation. Though such an approach is effective for several tasks, it has the similar limitation to model-based planning where the performance is highly dependent on the demonstration. Particularly, there is a work [7] that manages to deal with deformable object manipulation with reinforcement learning. However, it also employs imitation from humans as the demonstration.

In this paper, we tackle the DLO manipulation problem with a model-based reinforcement learning (RL) framework. Such a model is able to model DLO configurations directly without the need for demonstration. For simplicity, we concentrate on 2-D manipulation space. In RL, the problem is getting an agent, such as a robot manipulator to act in or on the world (e.g. by manipulating a DLO), so as to minimize its cost. Thus, the robot manipulation of a DLO with RL is formalized as follows: a DLO configuration that is stochastically modeled with inputs and outputs. Where inputs are actions sent from the robot and outputs are the new DLO configurations resulting from actions, and the costs that are given to the robot. Since RL enables the robot to plan actions which minimize the expected sum of cost, there is no expert knowledge required to assume the DLO model in advance. As a trade-off, the crucial problem for applying RL on DLO manipulation is data efficiency. Since a DLO has numerous configurations, inefficient learning means endless policy searching in the huge state space which will never converge. To address this problem, Probabilistic Inference for Learning Control (PILCO) is applied to the

[1] H. Han and T. Matsubara are with Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Nara, Japan

[2] G. Paul is with Centre for Autonomous Systems, University of Technology, Sydney, Australia

DLO manipulation, which is a sample-efficient model-based RL algorithm.

The contributions of this paper are the following:

- we present a model-based reinforcement learning approach for DLO manipulation without human-aided demonstration
- we validate our approach by performing a rope manipulation task in 2-D space

The remainder of this paper is organized as follows, Section II presents the related work. Section III presents the PILCO framework, and then details how to apply PILCO to DLO manipulation by employing the location-based point chain model. Section IV presents details of the experimental setup. Section V presents the experimental results. Section VI summarizes our work, and discusses future work.

II. RELATED WORK

Robot manipulation of a DLO is conventionally implemented by model-based planning. There have been several works that modeled DLO from different views. M.Saha and P. Isto [4] modeled DLO by its topological structure. S. Javdani et. al. [8] focused on an energy model. On the other hand, W. H. Lui and A. Saxena [5] focused on the model given by the features from a RGB-D sensor. More fundamentally, N. Alvarez et. al. [3] considered modeling from the DLOs' physical properties and the estimated physical parameters in the simulation engine. The review and comparison of the DLO model is beyond the scope of this paper, the reader can refer to [9] for the detail. However, since all the existing literature above focuses on modeling DLOs with some scenario-depended expert knowledge, the applications are restricted.

On the other hand, machine learning has been introduced to deal with deformable object manipulation problems by adapting different scenarios. A. X. Lee et. al. [6] employed machine learning for registering demonstrations to a new situation. More precisely, at first, multiple demonstrations were shown to the robot. In a new situation, the robot combined geometric warping with statistical learning to compute target configurations from demonstrations. The robot could adapt to the new situations by registering each of the demonstrations. Since the robot learnt from demonstrations, which came from either teleoperation or kinesthetic teaching, there was no relation to deformable object modeling. Hence the performance of this work was highly dependent upon human-aided demonstration. Similarly, B. Balaguer and S. Carpin [7] presented work that utilized reinforcement learning to exploit the mechanism of human imitation. Discrete points are employed in [7] to represent the appearance of the deformable object. Human imitation was a vital component of data sampling in order to overcome the high dimensionality of the deformable objects. Therefore, modeling and efficiently sampling data are the keys to apply machine learning to the DLO manipulation problem.

To address these problems and limitations, we propose a model-based reinforcement learning approach for DLO manipulation. Inspired by the model-free to model-based reinforcement learning comparison in [10], and the success in other domains [11] [12], we consider PILCO [1] as a sample-efficient framework with a point-based object representation [13] [14] based on location.

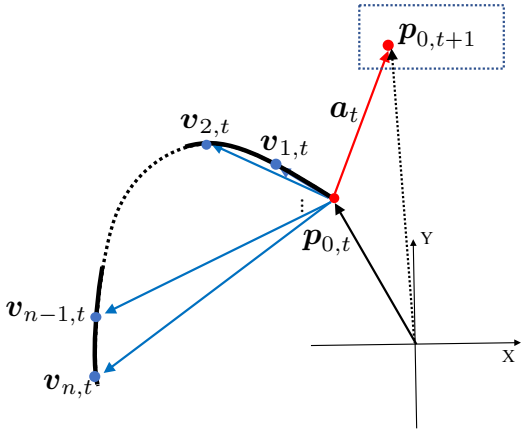

Fig. 2: Location-based point chain model for DLO. At step t the DLO configuration is specified by the absolute location of the first control point $\boldsymbol{p}_{0,t}$, and the relative location between first control point to others as $\boldsymbol{v}_{i,t}, i = 1,2...,n$. The action is the movement from $\boldsymbol{p}_{0,t}$ to $\boldsymbol{p}_{0,t+1}$.

III. MODEL-BASED REINFORCEMENT LEARNING FOR DLO MANIPULATION

This section presents our model-based RL approach for DLO manipulation. In Section III.A, we summarize the PILCO framework. Then, in Section III.B, we show how to apply it for DLO manipulation.

A. Probabilistic Inference for Learning Control

Probabilistic Inference for Learning Control (PILCO) has been investigated as a practical model-based policy search method in RL that works iteratively. Considering a dynamic system

$$\begin{aligned} \boldsymbol{s}_{t+1} &= f(\boldsymbol{s}_t, \boldsymbol{a}_t) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \boldsymbol{\Sigma}), \\ \boldsymbol{a}_t &= \pi(\boldsymbol{s}_t, \boldsymbol{\theta}), \end{aligned} \tag{1}$$

where continuous state, $\boldsymbol{s} \in \mathbb{R}^D$, action, $\boldsymbol{a} \in \mathbb{R}^F$, and unknown transition function, $f()$ with independent and identical distributed Gaussian noise, ε. In addition, action, $\boldsymbol{a}$ is the function of state, $\boldsymbol{s}$ with policy parameters, $\boldsymbol{\theta}$. In the PILCO framework, at each iteration, a probabilistic model of transition functions is given by the Gaussian Process from observed states, so that the policy search for $\boldsymbol{\theta}$ is performed to minimize the expected long-term cost, $J^{\pi}(\boldsymbol{\theta})$ based on the current model, where

$$J^{\pi}(\boldsymbol{\theta}) = \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{s}_t}[c(\boldsymbol{s}_t)], \boldsymbol{s}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \tag{2}$$

is given by T steps, and the cost, $c(\boldsymbol{s}_t)$ of being in state, x at step, t. Note after performing $\boldsymbol{a}_t$, the transition function model will be improved by observing the new state, $\boldsymbol{s}_{t+1}$ so as to improve the policy for the next action. As this iterative

learning process continues, PILCO will accomplish the task in the end.

1) Gaussian Process: Gaussian Processes (GP) is a nonparametric model for inferring an unknown function $f()$, where $y = f(\boldsymbol{x}) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, denoted as $f \sim \mathrm{GP}(m(.), k(.,.))$. Note that the mean function, $m(.)$ and the semi-positive-defined kernel function, $k(.,.)$ need to be clarified. In this paper, we use the mean function, $m(.) = 0$ and the Gaussian kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \delta_f^2 \exp\left(\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^\intercal \boldsymbol{M}(\boldsymbol{x}_i - \boldsymbol{x}_j)\right) + \delta_{\boldsymbol{x}_i \boldsymbol{x}_j}\sigma_n^2 \boldsymbol{I}, \quad (3)$$

where $\delta_{\boldsymbol{x}_i \boldsymbol{x}_j}$ is Kronecker delta, and the Hyper-parameters are $\boldsymbol{\theta}_{\mathrm{GP}} = [\delta_f^2, \boldsymbol{M}, \sigma_n^2]$. Given size-$n$ training set, $\mathscr{D} = (\boldsymbol{X}, \boldsymbol{y})$ where $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]^\intercal, \boldsymbol{y} = [y_1, y_2, ..., y_n]^\intercal$, $\boldsymbol{\theta}_{\mathrm{GP}}$ are optimized by maximizing marginal likelihood with automatic relevance determination [15].

Given a new input, $\boldsymbol{x}_*$, note matrix $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\boldsymbol{k}_* = k(\boldsymbol{X}, \boldsymbol{x}_*)$, GP predicts the output distribution as $y_* \sim \mathcal{N}(\mathbb{E}_f[y_*], var_f[y_*])$, where

$$\begin{aligned} \mathbb{E}_f[y_*] &= \mathbb{E}_f[f(\boldsymbol{x}_*)] = \boldsymbol{k}_*^\intercal (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}, \\ var_f[y_*] &= var_f[f(\boldsymbol{x}_*)] \\ &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^\intercal (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*). \end{aligned} \quad (4)$$

PILCO employs GP to model the transition function, $f()$ with input, $\boldsymbol{x}_t = [\boldsymbol{s}_t^\intercal, \boldsymbol{a}_t^\intercal]^\intercal, \boldsymbol{x}_t \in \mathbb{R}^{D+F}$, and corresponding training target, $\boldsymbol{y}_t = \boldsymbol{s}_{t+1} - \boldsymbol{s}_t, \boldsymbol{y}_t \in \mathbb{R}^D$. Note here conditionally independent GPs are trained for each dimension of $\boldsymbol{y}_t$. Thus, the one-step prediction model for eq. (1) is given by the posterior distribution of $\boldsymbol{s}_{t+1}$ as

$$\Pr(\boldsymbol{s}_{t+1} | \boldsymbol{x}_t) \sim \mathcal{N}(\boldsymbol{s}_t + \mathbb{E}_f[\boldsymbol{y}_t], var_f[\boldsymbol{y}_t]). \quad (5)$$

2) Moment Matching: Moment matching is an analytic approach for GP prediction with uncertain input, $\boldsymbol{x}_* \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}_*}, \boldsymbol{\Sigma}_{\boldsymbol{x}_*})$. The output distribution is predicted as $y_* \sim \mathcal{N}(\mu_{y_*}, \sigma_{y_*}^2)$, where $\mu_{y_*}, \sigma_{y_*}^2$ are computed from eq. (4) as

$$\begin{aligned} \mu_{y_*} =& \mathbb{E}_{\boldsymbol{x}_*}[\mathbb{E}_f[f(\boldsymbol{x}_*)] | \boldsymbol{\mu}_{\boldsymbol{x}_*}, \boldsymbol{\Sigma}_{\boldsymbol{x}_*}], \\ \sigma_{y_*}^2 =& \mathbb{E}_{\boldsymbol{x}_*}[\mathbb{E}_f^2[f(\boldsymbol{x}_*)] | \boldsymbol{\mu}_{\boldsymbol{x}_*}, \boldsymbol{\Sigma}_{\boldsymbol{x}_*}] + \mathbb{E}_{\boldsymbol{x}_*}[var_f[f(\boldsymbol{x}_*)] | \boldsymbol{\mu}_{\boldsymbol{x}_*}, \boldsymbol{\Sigma}_{\boldsymbol{x}_*}] \\ & - \mathbb{E}_{\boldsymbol{x}*}[\mathbb{E}_f[f(\boldsymbol{x}*)] | \boldsymbol{\mu}_{\boldsymbol{x}*}, \boldsymbol{\Sigma}_{\boldsymbol{x}_*}]^2. \end{aligned} \quad (6)$$

PILCO takes advantage of moment matching by regarding multiple steps prediction as the cascade of one-step predictions from the initial state, $\boldsymbol{s}_0$. Assume $\boldsymbol{s}_t$ is Gaussian distributed, since $\boldsymbol{a}_t$ is the function of $\boldsymbol{s}_t$, the joint distribution of $\boldsymbol{x}_t$ where $\Pr(\boldsymbol{x}_t) = \Pr(\boldsymbol{s}_t, \boldsymbol{a}_t)$ can be approximated as Gaussian. Substituting $\boldsymbol{x}_* = \boldsymbol{x}_t$ into eq. (6), then the prediction of $\boldsymbol{s}_{t+1}$ is given by eq. (5)-(6) as $\Pr(\boldsymbol{s}_{t+1} | \boldsymbol{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$, where

$$\begin{aligned} \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_{\boldsymbol{y}_t} + \boldsymbol{\mu}_t, \\ \boldsymbol{\Sigma}_{t+1} &= \boldsymbol{\Sigma}_t + \boldsymbol{\Sigma}_{\boldsymbol{y}_t} + \mathrm{cov}(\boldsymbol{x}_t, \boldsymbol{y}_t) + \mathrm{cov}(\boldsymbol{y}_t, \boldsymbol{x}_t), \end{aligned} \quad (7)$$

$\boldsymbol{\mu}_{\boldsymbol{y}_t}$ and $\boldsymbol{\Sigma}_{\boldsymbol{y}_t}$ is the multivariate case of μ_{y_*} and $\sigma_{y_*}^2$ separately as $\boldsymbol{x}_t = \boldsymbol{x}_*$ in eq. (6). Note that $\mathrm{cov}(\boldsymbol{x}_t, \boldsymbol{y}_t)$ and $\mathrm{cov}(\boldsymbol{y}_t, \boldsymbol{x}_t)$ can be analytically computed for the given policy. Thus, for the given policy, by assuming the initial input, $\boldsymbol{s}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, PILCO is able to compute the posterior distribution of $\boldsymbol{s}_t$ for any step, t by employing eq. (7) recursively.

3) Policy Improvement: PILCO improves the policy by optimizing policy parameters, $\boldsymbol{\theta}$ to minimize $J^\pi(\boldsymbol{\theta})$ in eq. (2). From eq. (7), the expected cost at step t is given by

$$\mathbb{E}_{\boldsymbol{s}_t}[c(\boldsymbol{s}_t)] = \int c(\boldsymbol{s}_t) \mathcal{N}(\boldsymbol{s}_t | \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \quad (8)$$

By choosing cost function, $c()$ so that eq. (8) can be computed analytically, the long-term cost, $J^\pi(\boldsymbol{\theta})$ can be optimized with respect to the policy parameters, $\boldsymbol{\theta}$ analytically by the gradient-based policy search, where the gradient is given by

$$\begin{aligned} \frac{\partial J^\pi(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \sum_{t=1}^{T} \frac{\partial \mathbb{E}_{\boldsymbol{s}_t}[c(\boldsymbol{s}_t)]}{\partial \boldsymbol{\theta}}, \\ \frac{\partial \mathbb{E}_{\boldsymbol{s}_t}[c(\boldsymbol{s}_t)]}{\partial \boldsymbol{\theta}} &= \frac{\partial \mathbb{E}_{\boldsymbol{s}_t}[c(\boldsymbol{s}_t)]}{\partial \boldsymbol{\mu}_t} \frac{d\boldsymbol{\mu}_t}{d\boldsymbol{\theta}} + \frac{\partial \mathbb{E}_{\boldsymbol{s}_t}[c(\boldsymbol{s}_t)]}{\partial \boldsymbol{\Sigma}_t} \frac{d\boldsymbol{\Sigma}_t}{d\boldsymbol{\theta}}. \end{aligned} \quad (9)$$

B. PILCO Application in DLO Manipulation

We apply the PILCO framework to DLO manipulation within 2D space as shown in Fig. 2. Firstly, we define the system state in eq. (1) with a location-based point chain model that specifies the DLO configuration. Then we define the action in eq. (1) as the movement of the end of the DLO. Lastly, a Gaussian-shaped cost function is given so that the integral of eq. (8) can be computed analytically.

State: We define the state with a location-based point chain model to depict the DLO configuration. More precisely, we approximate the shape of the DLO by n sequence control points, $\{p_0, p_1, ..., p_n\}$ in the XY-plane as shown in Fig. 2. The Cartesian coordinate for p_i at step, t is $\boldsymbol{p}_{i,t} = [x_{i,t}, y_{i,t}]^\intercal, i \in \{0, 1, 2, ..., n\}$. Note that p_0 is the control point that represents a DLO end. Initially, the DLO is defined as a straight line, such that all control points are evenly distributed, i.e. $\boldsymbol{p}_i - \boldsymbol{p}_{i-1} = \boldsymbol{p}_j - \boldsymbol{p}_{j-1}, \boldsymbol{p}_i - \boldsymbol{p}_j = (i-j)(\boldsymbol{p}_i - \boldsymbol{p}_{i-1}), i, j \in \{1, 2, ..., n\}$. Thus, the state at step t is defined by control points as

$$\boldsymbol{s}_t = [\boldsymbol{p}_{0,t}^\intercal, \boldsymbol{v}_{1,t}^\intercal, \boldsymbol{v}_{2,t}^\intercal ..., \boldsymbol{v}_{n-1,t}^\intercal, \boldsymbol{v}_{n,t}^\intercal]^\intercal, \quad (10)$$

where $\boldsymbol{v}_{i,t} = \boldsymbol{p}_{i,t} - \boldsymbol{p}_{0,t}, i = 1, 2, ..., n$.

Action: We focus on DLO manipulation by moving the end control point P_0 within 2-D space as shown in Fig. 2. Note that the DLO configuration varies with the movement of P_0. Thus, we define the action of step, t as

$$\boldsymbol{a}_t = (\boldsymbol{p}_{0,t+1} - \boldsymbol{p}_{0,t}). \quad (11)$$

Cost: In order to compute eq. (8) analytically, we employ a Gaussian-shaped cost function as

$$c(\boldsymbol{s}) = 1 - \exp\big(-\tfrac{1}{2\sigma_c^2} \|\boldsymbol{s} - \boldsymbol{s}_{\mathrm{target}}\|_{\boldsymbol{W}}^2\big), \quad (12)$$

where $\|\boldsymbol{s} - \boldsymbol{s}_{\mathrm{target}}\|_{\boldsymbol{W}}^2 = (\boldsymbol{s} - \boldsymbol{s}_{\mathrm{target}})^T \boldsymbol{W} (\boldsymbol{s} - \boldsymbol{s}_{\mathrm{target}})$ is the Mahalanobis distance between the current state and the target state. Matrix $\boldsymbol{W}$ weights correspondency for cost computation. The scaling parameter, σ_c^2 controls the width of the cost function.

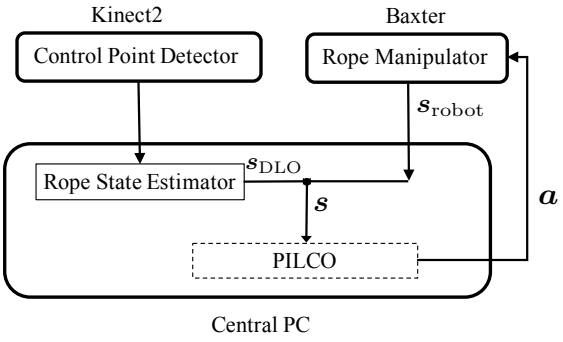

Fig. 3: The rope manipulation system consists of a Control Point Detector, a Rope Manipulator and a Central PC. Control Point Detector locates the control points; Rope Manipulator is the robot that manipulates the rope; Central PC is responsible for state generation, policy searching, and robot control. The dotted box represent policy search by PILCO.

Learning procedure: There are two phases in the learning process: the initial phase and the learning phase. In the initial phase, the robot performs random actions, and records corresponding DLO configurations to generate a training set. Next in the learning phase, firstly, a DLO model is generated by GP from the given training set; then, a policy search is performed based on the generated model to minimize the long-term cost; lastly, the robot takes actions according to the policy and current DLO configuration, and adds result configurations to the training set for the next iteration. Note that the learning phase continues iteratively with both updates of the model and policy. Finally, the learning process terminates when the prediction cost converges.

IV. EXPERIMENT

A. Rope Manipulation System

We built a rope manipulation system using the Robot Operating System (ROS). As illustrated in Fig. 3, there are three main components in the system: Control Point Detector, Rope Manipulator, and Central PC. In preparation, we mark the control points of the rope with two colors separately, where green is for p_0, and blue is for $p_i, i \neq 0$. The Control Point Detector captures control points by tracking markers, and sends the location information to the Central PC. In the Central PC, the system state is generated from the rope state, $\boldsymbol{s}_{DLO}$ and the robot state, $\boldsymbol{s}_{robot}$. The details are discussed in Section IV-C

In our experiment, a Baxter, dual-arm industry robot, acted as the Rope Manipulator. While a Kinect2 sensor was used as the Control Point Detector based upon a method from [16]. A workstation took the role of the Central PC that was responsible for state generation, policy searches and robot control.

B. Task

In our experiment, Baxter managed a 1-meter rope from the initial configuration to the target configuration. Note we represented the rope with 10 control points. For simplicity, we set rope manipulation in 2-D space as illustrated in Fig. 4. We set the task for Baxter to manipulate the rope so that the rope's two ends approach each other. Note there are two challenges for accomplishing the task. The first one comes from the high dimensionality of the DLO, since the rope has various configurations in the 2-D space. The second one comes from the manipulation system, since the robot has joint limits, and the Kinect2's viewpoint is also fixed, we introduced a robot cost to constrain Baxter's movement. Thus, the optimal policy is to minimize the long-term cost that includes robot cost.

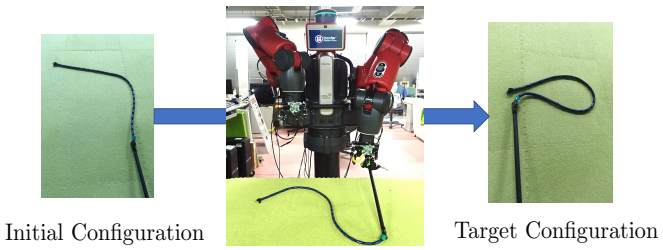

Fig. 4: Target configuration task. Baxter attempts to manipulate the rope from initial configuration to a target configuration within the green constraint area.

C. Setting

State: As illustrated in Fig. 3, the Rope State Estimator estimates the rope state based on the observation from Control Points Detector. With an assumption that the neighbor control points act similarly. The Rope State Estimator could determine each control point's location automatically by comparing the difference between current observation and previous rope state. Note that the Rope State Estimator requires control points location for initialization.

As shown in Fig. 4, we constrained the Baxter movement into the manipulation 2-D space by introducing the robot state. In the PILCO framework, system protection was considered by a constrained input signal, where exceeding the upper bound would lead to an empty rollout. This pre-setting helps to protect the system by stopping sampling. However, for the sake of efficiency in the early learning phase, where policy will lead to unstable actions due to the undeveloped modeling, we considered this by introducing the robot state to restrict the robot movement. Thus, out of range movements would partly be avoided due to the considerably high robot cost, and it would also speed up the sampling time.

As presented above, in our experiment, the state was generated by the Rope State Estimator as well as the Baxter robot. By considering robot state as the distance from its home position, we could easily define the robot state in our case by considering the gripper rotation and translation as in Fig. 5. As discussed in Section III-B, in our experiment we combined robot state and defined state as:

$$\boldsymbol{s}_t = [l_0, \theta_0, (\boldsymbol{p}_1 - \boldsymbol{p}_0)^\intercal, ..., (\boldsymbol{p}_n - \boldsymbol{p}_0)^\intercal, d]^\intercal, \qquad (13)$$

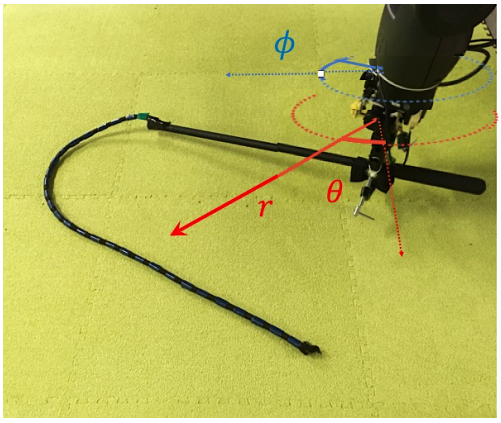

Fig. 5: Definition of robot action. A Baxter gripper manipulates a rope with a handle in 2-D space. The action of the gripper is defined by translation $[r, \theta]$ and rotation ϕ.

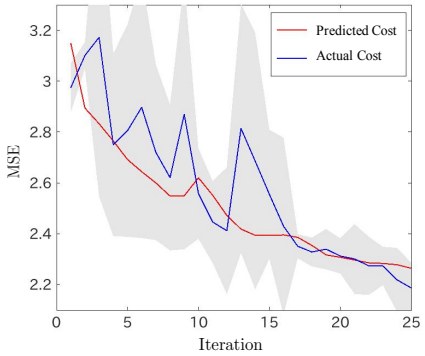

Fig. 6: Learning curve of the experimental result. X-axis is the learning iteration, Y-axis is the mean squared error of three experiments. The red curve is the mean prediction of the long-term cost from PILCO, while the blue line is the actual cost. The shaded part is for the standard deviation.

where $\boldsymbol{s}_{\text{robot}}$ is given by $[\theta_0, d]^\intercal$ such that θ_0 specifies the gripper's rotation, and d specifies the gripper's relative distance. $\boldsymbol{s}_{\text{DLO}}$ is given by $[l_0, \theta_0, (\boldsymbol{p}_1 - \boldsymbol{p}_0)^\intercal, ..., (\boldsymbol{p}_n - \boldsymbol{p}_0)^\intercal]^\intercal$ that the location of p_0 is defined by its position relative to the gripper's home position as the polar coordinate, $[l_0, \theta_0]$. Note that s_{robot} introduces an additional cost so that Baxter accomplishes the task within the desired area.

Action: We focus on rope manipulation by moving the end control point, P_0 within the 2D manipulation plane. Note that the rope configuration varies with the movement of P_0. In order to achieve different rope states by Baxter manipulation sufficiently, we considered two actions of Baxter's gripper within the XY-plane: translation and rotation, as illustrated in Fig. 5. With identical velocity settings, the action at step t is defined in the gripper's frame as

$$\boldsymbol{a}_t = [r_t, \theta_t, \phi_t]^\intercal, \tag{14}$$

In our setup, to avoid occlusion in control point detection, the Baxter's gripper manipulates a rope by a 0.5-meter handle. By fixing one end of the handle to the gripper, and the other end to the rope, the handle can be considered as an extension of the gripper as illustrated in Fig. 5. Note with this setup, p_0 was the junction between the handle and the rope. Given an action commander in the Central PC, the gripper translation, specified by r, θ in eq. (14), was implemented by computing the inverse kinematics of the Baxter arm's joints, while the gripper rotation, specified by ϕ in eq. (14), was implemented by rotating the wrist roll joint. Note that the gripper rotation led p_0 to make a circular motion within the constraint area. To constrain the movement of p_0 so that the limitations of the corresponding joints are not violated, we constrained $r \in [-0.09, 0.09], \phi \in [-\pi/2, \pi/2]$, where the units are meters and radians, respectively. In particular, such a constraint was considered as a constrained control signal in [1].

Cost: The benefit from the location-based point chain model is that we can conveniently design cost functions from eq. (12). For the experiment target configuration, we set $\boldsymbol{w} = diag(0, 0, ..., 1, 1, 0)$ and $\sigma_c = 0.5$ in eq. (12) separately. Note in our setup, PILCO optimized the long-term cost along with the robot state, such that

$$c(\boldsymbol{s}_t) = c_{\text{DLO}}(\boldsymbol{s}_{\text{DLO}}) + \alpha c_{\text{robot}}(\boldsymbol{s}_{\text{robot}}), \tag{15}$$

where c_{DLO} came from the rope state, while c_{DLO} came from the robot state, and the hyper-parameter, α scaled the contribution of the value of c_{DLO} for c. In our experiment, we set $\alpha = 0.2$.

V. Results

In order to validate our approach, we conducted three experiments with 25 iterations for each. In each iteration, Baxter took eight steps trying to manipulate the rope to the target configuration. Note in the initial phase of each experiment, Baxter generated a training set from 160 random actions.

All three experimental results are averaged as the learning results illustrated in Fig. 6. Since the actual cost is close to the predicted cost with a small standard deviation, the learning process efficiently captures the DLO model for achieving the task with robot movement constraints. More precisely, we illustrate snapshots in Fig. 7. At the early stage of learning, e.g. Iteration (Ite.) 1, Baxter uses the first two steps to drag the rope, and then place the two ends close to each other. However, the distance between the two ends is still large. As the learning process continues, e.g. Ite. 5, Baxter improves the policy for the first three steps for dragging. Finally, at the end stage learning, e.g. Ite. 25, Baxter employs the optimal policy which minimizes the sum of two ends' distance whilst considering the robot's movement constraint.

VI. Summary and Discussion

We have presented a model-based reinforcement learning approach to successfully manipulate DLOs without human aided demonstration. We conducted experiments to validate our approach by accomplishing a rope manipulation task in 2-D space.

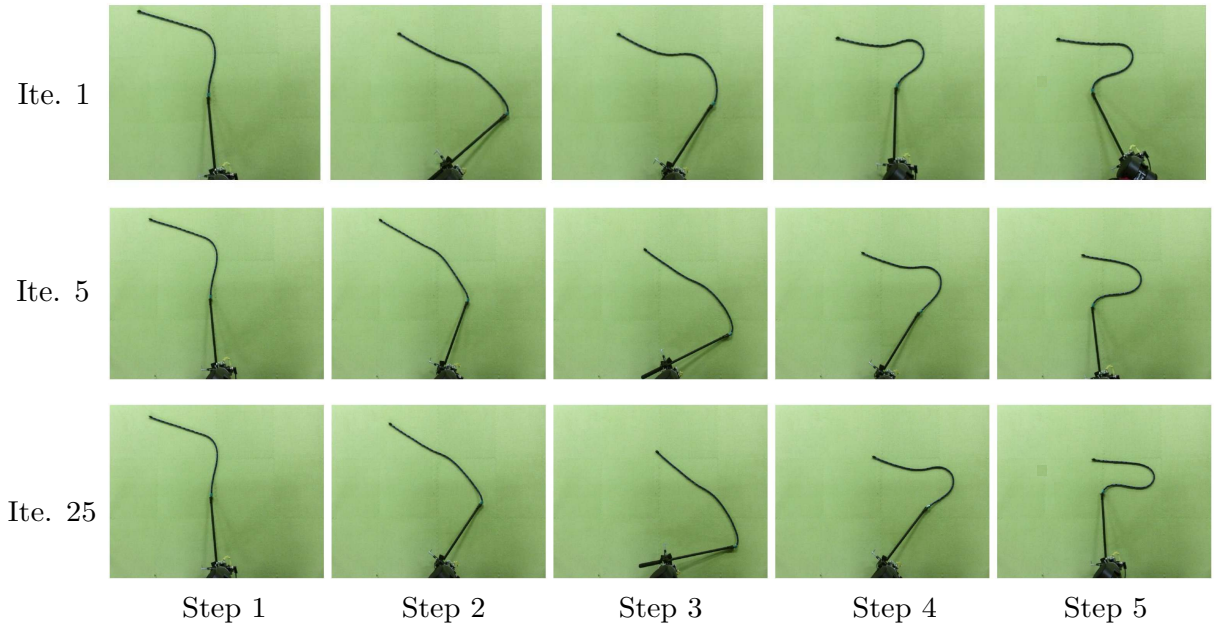

Fig. 7: Snapshots for the first-five steps of manipulation in iteration 1, 5, 25. As the number of iteration increased, the two end points of the rope are getting closer.

Several extensions for this work could be considered. Firstly, more complex tasks with a different DLO could be investigated by introducing more control points in the cost function. Secondly, the robot could manipulate the DLO directly without an extension handle. The challenges here include how to select the optimal manipulation position along the DLO, and how to avoid occlusions due to the gripper. Thirdly, more practical 3-D DLO manipulation could also be considered. The difficulties are how to expand the action space into 3-D space whilst considering the effect of gravity, and how to solve when the DLO is partially occluded by the arm, the gripper or the DLO itself.

ACKNOWLEDGMENT

We gratefully acknowledge the support from the New Energy and Industrial Technology Development Organization (NEDO) for this research.

REFERENCES

[1] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, Feb 2015.

[2] N. Kirchner, A. Alempijevic, S. Caraian, R. Fitch, D. Hordern, G. Hu, G. Paul, D. Richards, S. Singh, and S. Webb, "Robotassist-a platform for human robot interaction research," in *Proceedings of the Australasian conference on robotics and automation*. Australasian Conference on Robotics and Automation, 2010.

[3] N. Alvarez, K. Yamazaki, and T. Matsubara, "An approach to realistic physical simulation of digitally captured deformable linear objects," in *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Dec 2016, pp. 135–140.

[4] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1141–1150, Dec 2007.

[5] W. H. Lui and A. Saxena, "Tangled: Learning to untangle ropes with rgb-d perception," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, Nov 2013, pp. 837–844.

[6] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 177–184.

[7] B. Balaguer and S. Carpin, "Combining imitation and reinforcement learning to fold deformable planar objects," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, Sept 2011, pp. 1405–1412.

[8] S. Javdani, S. Tandon, J. Tang, J. F. O'Brien, and P. Abbeel, "Modeling and perception of deformable one-dimensional objects," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[9] P. Moore and D. Molloy, "A survey of computer-based deformable models," in *Proceedings of International Machine Vision and Image Processing Conference (IMVIP)*, Sept 2007, pp. 55–66.

[10] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, vol. 4, Apr 1997, pp. 3557–3564.

[11] M. Hamaya, T. Matsubara, T. Noda, T. Teramae, and J. Morimoto, "Learning assistive strategies from a few user-robot interactions: Model-based reinforcement learning approach," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3346–3351.

[12] B. Bischoff, D. Nguyen-Tuong, T. Koller, H. Markert, and A. Knoll, "Learning throttle valve control using policy search," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, ser. ECML PKDD 2013, vol. 8188. New York, NY, USA: Springer-Verlag New York, Inc., 2013, pp. 49–64.

[13] J. M. Saragih, S. Lucey, and J. F. Cohn, "Deformable model fitting by regularized landmark mean-shift," *Int. J. Comput. Vision*, vol. 91, no. 2, pp. 200–215, Jan. 2011.

[14] J. Takamatsu, T. Morita, K. Ogawara, H. Kimura, and K. Ikeuchi, "Representation for knot-tying tasks," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 65–78, Feb 2006.

[15] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[16] A. W. K. To, G. Paul, and D. Liu, "Surface-type classification using rgb-d," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 359–366, 2014.