2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Deformable Linear Object Prediction Using Locally Linear Latent Dynamics

Wenbo Zhang[1], Karl Schmeckpeper[2], Pratik Chaudhari[3], and Kostas Daniilidis[4]

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9560955

***Abstract*— We propose a framework for deformable linear object prediction. Prediction of deformable objects (e.g., rope) is challenging due to their non-linear dynamics and infinite-dimensional configuration spaces. By mapping the dynamics from a non-linear space to a linear space, we can use the good properties of linear dynamics for easier learning and more efficient prediction. We learn a locally linear, action-conditioned dynamics model that can be used to predict future latent states. Then, we decode the predicted latent state into the predicted state. We also apply a sampling-based optimization algorithm to select the optimal control action. We empirically demonstrate that our approach can predict the rope state accurately up to ten steps into the future and that our algorithm can find the optimal action given an initial state and a goal state.**

I. INTRODUCTION

Robotic manipulation can be applied to a variety of scenarios, such as surgery, housekeeping, manufacturing, and agriculture [1][2]. Deformable object manipulation is one of the important robotic manipulation problems. We face deformable objects frequently in our everyday lives, such as when folding the clothes [3], knot-typing [4][5], and moving a cable [6]. However, deformable object manipulation is still a challenging task due to the following reasons. (i) High model uncertainty. It is hard to simulate an accurate deformable object model because of environmental noises. (ii) Infinite-dimensional configuration spaces. We need to get lots of parameters for modeling a good deformable object. (iii) High cost of simulation. It is expensive to get all the simulation model parameters for deformable objects, and sometimes it is infeasible to have all the parameters. (iv) Hyper-underactuated systems. The robotic manipulator cannot fully control deformable objects with arbitrary trajectories. (v) Non-linear dynamics. Non-linearity makes it hard to predict future states when executing actions on deformable objects. Our paper explicitly addresses the problems caused by non-linear dynamics and infinite-dimensional configuration spaces through mapping dynamics from a non-linear space to a linear space and shrinking the state to a lower-dimensional latent state.

In this paper, we consider the problems on how to predict the future state of deformable linear objects (i.e., rope) given an initial state and a sequence of actions, and on how to find optimal control actions to move the rope from the initial state to a goal state given a prediction model.

To solve the prediction problem, we propose a prediction framework that consists of a state autoencoder model, an action autoencoder model, and a dynamics model. The state autoencoder model is based on convolutional neural networks. Rope dynamics are non-linear and are very complicated to learn directly. Unlike non-linear dynamics, linear dynamics are easier to learn and more efficient for state predictions. Thus, we consider mapping the non-linear space into a linear space, predicting future states in the linear space, and decoding back to the non-linear space from the linear space. To do so, we first use learned encoders to encode both the state and the action into the latent state and the latent action. Then, we run the dynamics model to get the state matrix and control matrix for the linear dynamics in the latent space. After that, we get a predicted next latent state by using locally linear latent dynamics based on the current latent state, latent action, state matrix, and control matrix. Finally, we decode the predicted next latent state into a predicted next state. Our experiments show that the prediction framework can accurately predict states up to ten timesteps in the future.

We also want to find the optimal control actions to manipulate the rope from an initial position to a goal position. To solve this control problem, we use a sampling-based optimization algorithm to select the optimal action sequence based on an initial state, a goal state, and the trained prediction framework. This algorithm is an extension of the cross-entropy method. The algorithm works by repeating the processes of (i) sampling action sequences from a probability distribution, (ii) predicting the goal state, (iii) calculating and sorting the loss between predicted state and ground truth, and (iv) fitting a new distribution to selected action sequences. We stop this process when the distribution converges and select the optimal action sequence with the lowest loss.

Contributions. Our key contributions are: (i) a novel prediction framework for predicting future states of deformable linear objects under a sequence of actions (Section III), (ii) a sampling-based optimization algorithm for selecting optimal control actions to move the deformable linear objects (Section III), (iii) experiments showing multi-step prediction results and comparing our method with a baseline qualitatively and quantitatively (Section IV), and (iv) ablation studies regarding locally and globally latent dynamics, model training methods, and how small latent size we can use to achieve good performance (Section IV).

[1]Wenbo Zhang did the work when he was a student at the GRASP Lab, University of Pennsylvania, USA zwenbo@seas.upenn.edu
[2]Karl Schmeckpeper is with the GRASP Lab, University of Pennsylvania, USA karls@seas.upenn.edu
[3]Pratik Chaudhari is with the GRASP Lab, University of Pennsylvania, USA pratikac@seas.upenn.edu
[4]Kostas Daniilidis is with the GRASP Lab, University of Pennsylvania, USA kostas@cis.upenn.edu
Code: https://github.com/zwbgood6/deform

II. RELATED WORK

Learning Models for Non-linear Systems. There are active studies about learning a model for non-linear dynamical systems. Kingma et al. [7] introduce the variational autoencoder (VAE) using approximate Bayesian inference. Based on VAE, Watter et al. [8] propose an Embed to Control (E2C) method to learn a non-linear dynamical system model. They use the VAE to learn latent parameters and set locally linear constraints on dynamics using an optimal control concept. We are inspired by this method. We do experiments using locally linear latent dynamics and compare it with globally linear latent dynamics. However, instead of applying the optimal control formulation, we train an action-conditioned dynamics model to enable system identification in the latent space. Our dynamics model can generate the state matrix and the control matrix at each time step. Li et al. [9] apply the Koopman operator theory to provide nonlinear-to-linear transformation. Based on this approach, the infinite-dimensional state space changes to finite-dimensional state space. We learn from this method to encode high-dimensional state space into a smaller latent state in the linear space. There are also many researchers [10][11][12] working on learning high-dimensional non-linear state space and system identification.

State Estimation and Prediction. For state estimation, Yan et al. [13] apply self-supervised learning for state estimation. Schulman et al. [14] incorporate the point cloud in the probabilistic generative model for tracking deformable objects. Petit et al. [15] track objects with point cloud data from an RGB-D camera. Unlike those approaches to getting the object's points for state estimation, we use the binary mask of rope image as an input state to a state autoencoder. Then, the encoder can output a latent state. For the prediction of state, Finn et al. [16][17] present deep action-conditioned video prediction models. Unlike [16][17], we predict the future states in the latent space based on the estimation of the latent states. Besides, we predict the next state using the current state information without including previous states'.

Planning and Control. The work [13] uses Model Predictive Path Integral (MPPI) controller for choosing the best grasping point with the least moving length to the goal. Wang et al. [18] apply the data-driven approach to generate a sequence of planned images and then use an inverse dynamics model to execute the plan. Sundaresan et al. [19] use dense depth object descriptors to learn a policy for manipulating a rope into different configuration settings. Pathak et al. [20] learn a goal-conditional control policy for the rope manipulation based on a sequence of images. Moll et al. [21] compute the minimal-energy curves for path planning in a wire manipulation task. Bayazit et al. [22] introduce a Probabilistic Roadmap (PRM) method. There are also some other sampling-based methods which have been introduced in [4][17][23][24][25]. Similar to [17], we have a similar control setup and we also use a sampling-based optimization algorithm to select the optimal control action. Besides, researchers apply imitation learning to learn control policies. Nair et al. [26] use human demonstrations to teach the robot how to move a rope. The robot will learn the policy from the video sequence and apply this policy for execution. Huang et al. [27] approach the problem by trajectory transfer through non-rigid registration.

Deformable Objects Manipulation. There are mainly two types of deformable objects. One is about linear objects [25] (e.g., rope, cable) and the other is about non-linear objects [3][28] (e.g., fabric, cloth, bed sheet, garment). This paper is focusing on a simpler scenario on deformable linear objects, such as the ropes. Besides, our attentions are more on how to predict the future states and how to sample actions to reach a goal state, and less on how to manipulate the deformable objects using robots.

III. METHOD

Deformable objects modeling [15][28][29][30] is a good way to model the objects in a simulation. However, existing methods are not practical for real-time robotic tasks due to slow speed, high simulation expense, high-dimensional state space, high model uncertainty, and non-linear dynamics. These reasons result in a challenging task of predicting deformable objects in future states under specific actions. Thus, we want to simplify the non-linear dynamics and reduce high-dimensional state space for deformable linear objects.

Instead of using a physics-based simulation to model the objects, we design an encoding network to extract the latent states from rope images. Then we learn the dynamical system in the latent space. Using the dynamics in the latent space, we can predict the future latent states. After that, the model decodes the future latent states into rope images. We elaborate on each step as following.

A. Preliminary

First, we introduce some fundamental concepts.

Rope Dynamics. We consider nonlinear rope dynamics

$$x(t+1) = f(x(t), u(t)),$$

where $t \in \{0, 1, ..., T\}$ is the time step, $x(t) \in \mathcal{X} \subseteq \mathbb{R}^m$ is the state, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^n$ is the control input. Since we cannot get the rope state easily, we denote the rope image as the state $x(t)$. In our case, $u = (x, y, l, \theta)$, where x and y are positions in the image space, moving length l and moving angle θ are in the world space.

Latent Dynamics. The rope dynamics are non-linear, very complicated to learn, and hard to generalize, while linear dynamics are generally simpler to learn and can help make state predictions more easily. Thus, we consider a linear dynamics model

$$g(t+1) = K(t)g(t) + L(t)a(t) \tag{1}$$

in the latent space. Here, $g(t) \in \mathcal{G} \subseteq \mathbb{R}^p$ is the latent state, $a(t) \in \mathcal{A} \subseteq \mathbb{R}^q$ is the latent action, $K(t) \in \mathcal{K} \subseteq \mathbb{R}^{p \times p}$ is the state matrix, and $L(t) \in \mathcal{L} \subseteq \mathbb{R}^{p \times q}$ is the control matrix. The state matrix $K(t)$ is a transition matrix from the current latent state $g(t)$ to the next latent state $g(t+1)$. The control

matrix $L(t)$ learns the impact of the latent action $a(t)$ on the state transition.

For globally linear latent dynamics, $K(t) = K$ and $L(t) = L$ for all $t \in \{0, 1, ..., T\}$, where K and L are fixed matrices. For locally linear latent dynamics, $K(t)$ depends on the state $x(t)$ and $L(t)$ depends on both state $x(t)$ and action $u(t)$.

Encoder Models. The state encoder model is a map $\phi_e : \mathcal{X} \to \mathcal{G}$, the action encoder model is a map $\varphi_e : \mathcal{U} \to \mathcal{A}$. We then have

$$g(t) = \phi_e(x(t)),\ a(t) = \varphi_e(u(t)).$$

Encoder models encode the state x into the latent state g and the action u into the latent action a. For the state x (i.e., raw image), it is high-dimensional and redundant, so we want to reduce the state size and use the smaller latent state as a representation. For the action u (i.e., grasping positions, gripper moving length, and gripper moving angle), the raw action usually has a very non-linear effect on the state, so we lift the actions to a higher dimension and make them easier to linearize in the latent space.

Decoder Model. The state decoder model is a map $\phi_d : \mathcal{G} \to \mathcal{X}$, the action decoder model is a map $\varphi_d : \mathcal{A} \to \mathcal{U}$. We then have

$$\hat{x}(t) = \phi_d(g(t)),\ \hat{u}(t) = \varphi_d(a(t)).$$

Paired with encoder models, decoder models decode the latent state g into the reconstructed state $\hat{x}$ and the latent action a into the reconstructed action $\hat{u}$.

Dynamics Model. The dynamics model contains one map $\psi_s : \mathcal{X} \to \mathcal{K}$ for the state matrix and the other map $\psi_c : \mathcal{X},\mathcal{U} \to \mathcal{L}$ for the control matrix. We then have

$$K(t) = \psi_s(x(t)),\ L(t) = \psi_c(x(t), u(t)).$$

B. Prediction Framework

We consider the prediction framework in Figure 1. It consists of the state autoencoder model, action autoencoder model, and dynamics model.

When predicting, given a new state $x(t)$ and a new action $u(t)$, we can get latent state $g(t)$, latent action $a(t)$, state matrix $K(t)$, control matrix $L(t)$. Using locally linear latent dynamics, we can have predicted next latent state $g^{pred}(t+1) = K(t)g(t) + L(t)a(t)$. By decoding the predicted next latent state $g^{pred}(t+1)$, we will have the predicted next state $x^{pred}(t+1)$.

Loss Function. We consider minimizing the losses for state autoencoder model, action autoencoder model, and dynamics model. The loss of state autoencoder model is

$$\mathcal{L}_{state} = \sum_{t=0}^{T} ||\phi_d(\phi_e(x(t))) - x(t)||_2^2. \quad (2)$$

The state loss $\mathcal{L}_{state}$ makes sure the reconstructed state $\hat{x}$ is similar to ground truth state x. The loss of action autoencoder model is

$$\mathcal{L}_{action} = \sum_{t=0}^{T} ||\varphi_d(\varphi_e(u(t))) - u(t)||_2^2. \quad (3)$$

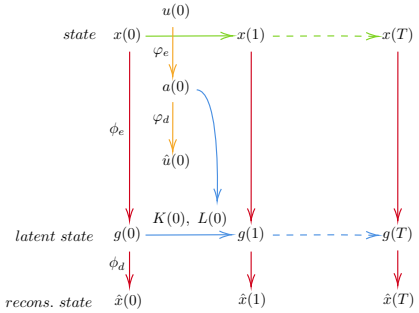

Fig. 1: Diagram for prediction framework. We take a part of the diagram from time step 0 to 1 as an example. The red arrows show the state autoencoder model. We use a state encoder ϕ_e to encode a state $x(0)$ into a latent state $g(0)$ and then use a state decoder ϕ_d to decode the latent state $g(0)$ into a reconstructed state $\hat{x}(0)$. The orange arrows show the action autoencoder model. We use an action encoder φ_e to encode an action $u(0)$ into a latent action $a(0)$ and then use an action decoder φ_d to decode the latent action $u(0)$ into a reconstructed action $\hat{u}(0)$. The green arrows show the rope dynamics. Given the state $x(0)$ and the action $u(0)$, we can have next state $x(1)$. The blue arrows show the latent dynamics. Given the latent state $g(0)$, the latent action $a(0)$, state matrix $K(0)$, and control matrix $L(0)$, we can get the next latent state $g(1)$ based on locally linear latent dynamics.

The action loss $\mathcal{L}_{action}$ makes sure the reconstructed action $\hat{u}$ is similar to ground truth action u. The loss of dynamics model is

$$\mathcal{L}_{dyn} = \sum_{t=0}^{T} ||K(t)g(t) + L(t)a(t) - g(t+1)||_2^2. \quad (4)$$

The dynamics loss $\mathcal{L}_{dyn}$ sets a locally linear constraint on latent dynamics. The loss of prediction model is

$$\mathcal{L}_{pred} = \sum_{t=0}^{T-1} ||x^{pred}(t+1) - x(t+1)||_2^2. \quad (5)$$

The prediction loss $\mathcal{L}_{pred}$ makes sure the predicted state x^{pred} is similar to ground truth state x. The overall training loss is

$$\mathcal{L} = \mathcal{L}_{state} + \lambda_1 * \mathcal{L}_{action} + \lambda_2 * \mathcal{L}_{dyn} + \lambda_3 * \mathcal{L}_{pred},$$

where λ_1, λ_2, and λ_3 are coefficients.

Training Prediction Framework. Figure 2 shows 4 steps on how to train the prediction framework $\mathcal{M}$.

Training Autoencoders. In Figure 2 Step 1, we use Equation (2) and Equation (3) to train the autoencoders to get both latent states g and latent actions a. When state loss $\mathcal{L}_{state} < \epsilon_{state}$ and action loss $\mathcal{L}_{action} < \epsilon_{action}$, we save latent state g and latent action a in each time step. Here, ϵ_{state} and ϵ_{action} are very small values. We make sure state x and action u are similar to reconstructed state $\hat{x}$ and reconstructed action $\hat{u}$, respectively. We also do early stopping to prevent the autoencoders from overfitting to the training set.

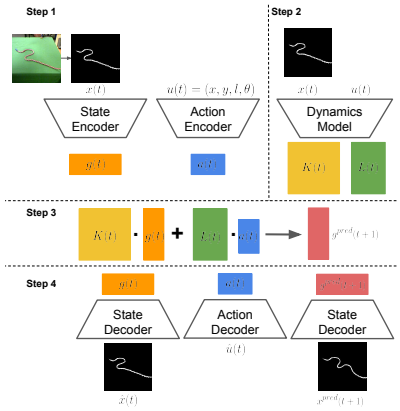

Fig. 2: Prediction framework in four steps. (i) Step 1: we do color space segmentation to get a binary mask from the raw rope image. We use this mask as the input $x(t)$ for the state encoder and the output is the latent state $g(t)$ (orange vector). We also use action encoder to encode the action $u(t)$ into the latent action $a(t)$ (blur vector). (ii) Step 2: we train the dynamics model using the binary mask $x(t)$ and the action $u(t)$. The output is state matrix $K(t)$ (yellow matrix) and control matrix $L(t)$ (green matrix). (iii) Step 3: by using locally linear latent dynamics, we have predicted next latent state $g^{pred}(t+1) = K(t)g(t) + L(t)a(t)$. (iv) Step 4: we use the state decoder to decode the latent state $g(t)$ into the reconstructed state $\hat{x}(t)$, run the action decoder to decode the latent action $a(t)$ into reconstructed action $\hat{u}(t)$, and apply the same state decoder to decode the predicted next latent state $g^{pred}(t+1)$ into predicted next state $x^{pred}(t+1)$.

Training Dynamics Model. The state matrix K depends on state x and the control matrix L depends on both state x and action u. In Figure 2 Step 2, we train dynamics model to minimize the dynamics loss $\mathcal{L}_{dyn}$ in Equation (4) and prediction loss $\mathcal{L}_{pred}$ in Equation (5).

There are two methods for model training: (i) training autoencoder models first and then training dynamics model, and (ii) training autoencoder models and dynamics model together. We discuss the model training order and our preference in Section IV ablation study.

To wrap up, after training the autoencoders and the dynamics model using the training set, we start to make predictions for new states and actions in the test set. The detailed architectures and hyperparameters are available in the code https://github.com/zwbgood6/deform.

C. Sampling-based Model Predictive Control

We consider using cross-entropy method (CEM) [31] to select optimal action given trained prediction framework $\mathcal{M}$. CEM is an optimization algorithm. It works by repeating two phases: (i) providing a probability distribution and sampling from this distribution and (ii) minimizing the cross-entropy between the current distribution and a target distribution for better sampling in the next iteration. Our Algorithm 1 is an extension of CEM. For each time step, given trained prediction framework $\mathcal{M}$, initial image x_{init}, and goal image x_{goal}, we first initialize a uniform distribution Q_{pre}. Then, we sample m action sequences from distribution Q_{pre}. Each action sequence has length h, meaning that there are h actions in this sequence. We use the prediction framework $\mathcal{M}$ to predict the future states x^{pred} given initial image x_{init} and each action sequence. Assuming we can get a predicted goal image x^{pred}_{goal} after taking h actions, we then calculate the Binary Cross Entropy (BCE) loss between the predicted goal images x^{pred}_{goal} and goal image x_{goal} we want. By sorting m BCE losses, we choose n $(n < m)$ action sequences that have lowest losses. We apply multivariate normal distribution Q_{post} to fit n samples. If the KL divergence between Q_{pre} and Q_{post} is not less than a specified small value, we will repeat the same process until the distribution Q converges. After distribution convergence, we execute the optimal action with the lowest loss.

Algorithm 1 Sampling-based optimization algorithm

Input prediction framework $\mathcal{M}$, initial image x_{init}, goal image x_{goal}
  for $t = 1, ..., T$ **do**
    Initialize Q with uniform distribution
    while Q does not converge **do**
      Sample m action sequences with length h from Q
      Use prediction framework $\mathcal{M}$ to predict goal images x^{pred}_{goal} using m action sequences
      Calculate loss between predicted goal image x^{pred}_{goal} and goal image x_{goal}
      Select n action sequences with lowest sorted losses
      Fit multi-variate normal distribution Q to n samples
    end while
    Execute action with lowest loss
    Observe new next state
  end for

IV. EXPERIMENTS AND RESULTS

We demonstrate that our prediction framework can predict well, and our model can be used for control. We also show that training methods and locally linear latent dynamics are critical for good performance.

A. Experimental Setup

We use Rope Manipulation Dataset from paper [26]. This dataset is challenging because (i) the rope is deformable and we do not know the rope dynamics beforehand, (ii) the images are not taken straight down and there is a less-than-ninety-degree angle between the camera and the table, and (iii) the consecutive actions are not applied on the same grasping position. These challenges combine to make it difficult to predict the rope state accurately at each time step.

State. Different from [26] which uses the whole image as an input, we merely want to get the rope-related information in the state. Thus, we do color space segmentation and add a black mask (Figure 2, Step 1) to replace the noisy background. We resize the image from 240×240 to 50×50 and

apply data augmentation consisting of translation, vertical flips, and horizontal flips.

Action. The original action $u = (x, y, l, \theta, \mathbb{1})$ includes x and y positions in the image space, moving length l and moving angle θ in the world space, and an indicator $\mathbb{1}$ that decides whether this action validly moves the rope or not. Situations for the invalid actions include the robotic gripper not contacting the rope and not moving the rope according to the command. We re-calculate the action x and y positions according to the change of image size and filter the invalid actions when training and testing.

B. Multi-step Prediction Results And Comparison

One-step Prediction. In Figure 2 Step 3, we use Equation (1) to get predicted next latent state g_{pred} based on the current state matrix K, the current latent state g, the current control matrix L, and the current latent action a. In Figure 2 Step 4, we use the same decoder in the state autoencoder and decode the predicted next latent state g_{pred} into predicted next state x_{pred}.

Multi-step Prediction. In order to test our framework's prediction horizon, we use the trained prediction model for multi-step prediction. Given state $x(0)$, action $u(0)$, and trained model $\mathcal{M}$, we can predict next state $x^{pred}(1)$. Then using predicted state $x^{pred}(1)$ and ground truth action $u(1)$, we predict the next state $x^{pred}(2)$. By iterating the same process, we get ten predicted states shown in Figure 3.

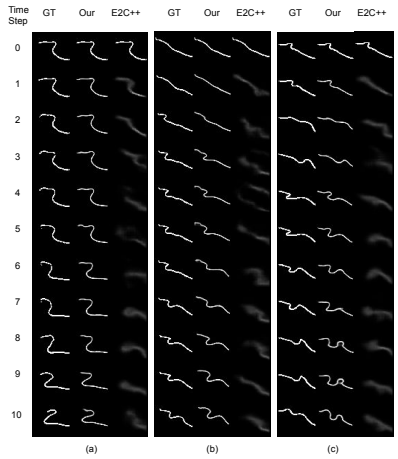

Fig. 3: Comparison between Ground Truth (GT), our method (Our), and E2C++ from time step 0 to 10. (a), (b) and (c) represent three cases starting at different initial states. In each case, it includes GT, Our, and E2C++. The first row is the initial state at time step 0. The second row (time step 1) to the last row (time step 10) are the predicted states. We can conclude that our method can predict the rope state accurately up to ten steps into the future, while the prediction results using baseline E2C++ are blurry.

Comparison. We compare our results with Ground Truth and results using Embedded to Control (E2C) [8]. E2C is a method for learning and control of non-linear dynamics from images. Unlike E2C using optimal control formulation in the latent space, we use the dynamics model to get the state matrix and the control matrix for state prediction in the latent space. To make a fair comparison, we use the same neural network architecture for autoencoders and the same latent size for the state. When running the E2C on the Rope Manipulation Dataset, we find the reconstruction and prediction results are blurry, and we cannot observe a clear rope shape from the image. Thus, we add a prediction loss $\mathcal{L}_{pred}$ and a state loss $\mathcal{L}_{state}$ on the original loss of variational bound and KL divergence. We name this new approach E2C++. Figure 3 (a) and (b) show that our method can predict the future states well in ten steps, and the rope shape is similar to the ground truth. For Figure 3 (c), there is a clear distinction between GT and our method at time step 8 because the previous states (time step 1 to 7) gradually accumulate the state prediction error. E2C++ can only predict a blurry rope shape within two-time steps. The remaining state predictions using E2C++ do not make sense.

C. Action Planning and Control Results

Qualitative Results. We use the sampling-based optimization method, shown in Algorithm 1, to plan rope manipulation. The objective is to generate an optimal action sequence. Similar to the standard cross-entropy method (CEM), our algorithm stops repeating in the inner loop when the probability distribution Q converges. However, unlike repeating two phases for the standard CEM, our algorithm needs to do more steps of sampling action sequences, predicting the goal state, calculating and sorting the loss, and fitting a new distribution. In the experiments, we first sample the optimal action by applying CEM. Then we use the trained prediction framework to predict the next state $x^{pred}(t+1)$ given the current state $x(t)$ and the sampled optimal action.

Figure 4 shows the experimental results. We overlay sampled action on the rope and compare it with the ground truth action. By comparing the Ground Truth (GT) action with sampled action using CEM, we can find sampled action is close to GT action concerning grasping positions, moving length, and moving angle. However, sometimes there are many possible actions to move the rope from one state to the next state. For example, Figure 4 (c) shows the different grasping positions for sampled action, but the rope ends up with a similar predicted next state as the GT state.

Quantitative Results. We calculate the Mean Square Error (MSE) between sampled actions and ground truth actions. We also apply MSE between the predicted next state and ground truth next state. After running 100 sampled one-step actions using our method and E2C++, we get the mean and standard deviation in Table I. By comparing Action Score, we can observe that our score is lower. It means the sampled action is closer to the ground truth action when using our prediction model. From State Score, we can find our method has a lower score. It means our method is better at predicting the next state after applying the sampled action.

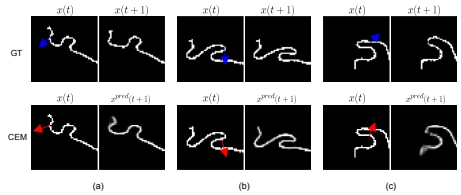

Fig. 4: Comparison between Ground Truth (GT) actions and sampled actions using the Cross-Entropy Method (CEM). (a), (b) and (c) are three cases with different initial rope positions. In each case, the first row shows GT action (blue arrow) at state $x(t)$ (left) and the GT next state $x(t+1)$ (right). The second row shows sampled action (red arrow) at state $x(t)$ (left) and the predicted next state $x^{pred}(t+1)$ (right). We can conclude that the sampled actions using CEM are close to GT actions concerning the grasping positions, the moving length, and the moving angle. For case (c), sometimes there are multiple options to move the rope. Besides, our prediction model can accurately predict the next state. The predicted state $x^{pred}(t+1)$ is similar to GT state $x(t+1)$.

Method	Action Score (MSE)	State Score (MSE)
Our	**18.20 ± 9.46**	**0.59 ± 0.27**
E2C++	29.35 ± 5.86	1.12 ± 0.18

TABLE I: Action score and state score comparison between our method and E2C++.

D. Ablation Study

Locally Latent Dynamics v.s. Globally Latent Dynamics. For locally latent dynamics, state matrix $K(t)$ and control matrix $L(t)$ are different at each time step. However, K and L are constant matrices for globally latent dynamics. If K and L are constant matrices, it will set a more strict constraint in the latent space. It is difficult to train a good-performance model with globally linear latent dynamics since we need to guarantee the dynamics are linear for all latent states. In our experiments, we use locally linear latent dynamics.

Order for Training Models. We have autoencoder models for state and action and the dynamics model. The order to train models is important to get good performance. We use two methods: (i) training autoencoder models first and then training dynamics model, (ii) training autoencoder models and dynamics model together. We demonstrate that method (i) can provide better prediction and control results, as shown in Figure 3, while method (ii) reconstructs blurry rope images, and we cannot find a clear rope shape. The reason is that we can get a good latent state estimation when training autoencoder models. After having a good latent state estimation, we then train the dynamics model to get good system dynamics. For method (ii), training all models together to optimize all losses is more difficult to get a good prediction result. It is because we need to find the optimal neural network weights to get accurate latent state, latent action, state matrix, and control matrix at the same time.

Latent State and Latent Action Size. Since the dimension of state x is 50×50, we need to shrink its size to a smaller dimension for latent state g. Different from the state, we use the autoencoder model to expand the action dimension from (x, y, l, θ) to a larger size. We do experiments to find how small we can get for the latent size to achieve good performance. Experiments show that the latent state size cannot be less than 50, and latent action size cannot be less than 10. When the latent state and latent action size are too small, there is no enough information about the real state and action. Besides, their sizes do not need to be too large (e.g., latent state size is 300+ and latent action size is 100+). One reason is that it increases more time to train since we have state matrix K and control matrix L as well. Another reason is that the larger size does not improve the prediction performance much. Thus, we choose 80 and 80 for latent state size and latent action size, respectively.

Whether Using Action Autoencoder. In the prediction framework, we use the autoencoder to lift the action space to a higher dimensional space so it can work linearly with the latent state in the latent space. We also try not to use the autoencoder in the prediction framework, and we cannot achieve good prediction results due to the non-linearity of the impact of the four-element action.

V. CONCLUSIONS

This paper presents a prediction framework for rope prediction. Our method first maps a non-linear space into a locally linear space using the encoders. Then, we get the state matrix and control matrix for linear dynamics by running the dynamics model. In the next step, we make a prediction based on the locally linear latent dynamics. Finally, we decode the predicted latent state into the predicted state in the non-linear space. We also propose a sampling-based optimization algorithm to select the optimal control action to move the rope from an initial state to a goal state. The experimental results demonstrate that our method can accurately predict the rope images to ten steps in the future. The experiments also show that our sampling-based optimization algorithm can find the optimal actions to relocate the rope. By comparing sampled actions using our method and actions using the baseline, we demonstrate that our method achieves better performance.

Our approach has not demonstrated success on more complex rope configurations for linear rope prediction, such as loops or knots. Future work includes predicting more complex rope configurations, generalizing the rope prediction framework across different rope materials, and deploying our framework on a robotic manipulator.

ACKNOWLEDGMENT

We thank Bernadette Bucher for her contributions to this project. The research was sponsored by the Army Research Office and was accomplished under grants ARO MURI W911NF-20-1-0080, NSF CPS 2038873, ARL DCIST CRA W911NF-17-2-0181, ONR N00014-17-1-2093, and by the Honda Research Institute.

REFERENCES

[1] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[2] F. F. Khalil and P. Payeur, "Dexterous robotic manipulation of deformable objects with multi-sensory feedback-a review," in *Robot Manipulators Trends and Development*. IntechOpen, 2010.

[3] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrik, A. Kargakos, L. Wagner, V. Hlaváč, T.-K. Kim, and S. Malassiotis, "Folding clothes autonomously: A complete pipeline," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1461–1478, 2016.

[4] M. Saha and P. Isto, "Motion planning for robotic manipulation of deformable linear objects," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2478–2484.

[5] W. Wang and D. Balkcom, "Tying knot precisely," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3639–3646.

[6] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, "Cable manipulation with a tactile-reactive gripper," *arXiv preprint arXiv:1910.02860*, 2019.

[7] D. P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder," in *Second International Conference on Learning Representations, ICLR*, vol. 19, 2014.

[8] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in neural information processing systems*, 2015, pp. 2746–2754.

[9] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba, "Learning compositional koopman operators for model-based control," *arXiv preprint arXiv:1910.08264*, 2019.

[10] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Thirty-first aaai conference on artificial intelligence*, 2017.

[11] M. Karl, M. Soelch, J. Bayer, and P. Van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," *arXiv preprint arXiv:1605.06432*, 2016.

[12] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4832–4839.

[13] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2372–2379, 2020.

[14] J. Schulman, A. Lee, J. Ho, and P. Abbeel, "Tracking deformable objects with point clouds," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1130–1137.

[15] A. Petit, V. Lippiello, and B. Siciliano, "Real-time tracking of 3d elastic objects with an rgb-d sensor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3914–3921.

[16] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, 2016, pp. 64–72.

[17] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.

[18] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar, "Learning robotic manipulation through visual planning and acting," *arXiv preprint arXiv:1905.04411*, 2019.

[19] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9411–9418.

[20] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 2050–2053.

[21] M. Moll and L. E. Kavraki, "Path planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 625–636, 2006.

[22] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Probabilistic roadmap motion planning for deformable objects," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 2126–2133.

[23] E. Anshelevich, S. Owens, F. Lamiraux, and L. E. Kavraki, "Deformable volumes in path planning applications," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 3. IEEE, 2000, pp. 2290–2295.

[24] F. Lamiraux and L. E. Kavraki, "Planning paths for elastic objects under manipulation constraints," *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 188–208, 2001.

[25] O. Roussel, A. Borum, M. Taix, and T. Bretl, "Manipulation planning with contacts for an extensible elastic rod by sampling on the submanifold of static equilibrium configurations," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3116–3121.

[26] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2146–2153.

[27] S. H. Huang, J. Pan, G. Mulcaire, and P. Abbeel, "Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 878–885.

[28] Y. Li, Y. Wang, Y. Yue, D. Xu, M. Case, S.-F. Chang, E. Grinspun, and P. K. Allen, "Model-driven feedforward prediction for manipulation of deformable objects," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1621–1638, 2018.

[29] S. F. Gibson and B. Mirtich, "A survey of deformable modeling in computer graphics," Citeseer, Tech. Rep., 1997.

[30] N. Essahbi, B. C. Bouzgarrou, and G. Gogu, "Soft material modeling for robotic manipulation," in *Applied Mechanics and Materials*, vol. 162. Trans Tech Publ, 2012, pp. 184–193.

[31] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.