Deformable Objects Manipulation Using The Koopman Operator

Wenbo Zhang

A Thesis

in

School of Engineering and Applied Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2020

Kostas Daniilidis
Ruth Yalom Stone Professor
Supervisor of Thesis

Pratik Chaudhari
Assistant Professor
Co-supervisor of Thesis

Camillo J. Taylor
Graduate Group Chair

© COPYRIGHT

2020

Wenbo Zhang

This work is licensed under the

Creative Commons Attribution

NonCommercial-ShareAlike 3.0

License

To view a copy of this license, visit

http://creativecommons.org/licenses/by-nc-sa/3.0/

To my loving parents and grandparents.

Acknowledgment

I thank my advisor Kostas Daniilidis for his support and guidance throughout my Master's. His profession exerts an imperceptible influence on me, which tells me to build the research work step by step and evaluate my idea quickly after implementation. His kindness and encouragement inspired me when I felt like I was stuck and going nowhere. He is also caring about students' well-being, for which I am truly grateful. I also thank my co-advisor Pratik Chaudhari for meeting with me every week, and revising my personal statement and giving me guidance on the universities when I applied for Ph.D. He always asks me a lot of questions about my ideas, which pushes me to rethink the idea's feasibility. Sometimes I would be amazed by his different perspective to solve a problem.

I thank Tianwen Tony Cai for motivating me to pursue the Ph.D. I thank Vijay Kumar for driving my curiosity of doing research in robotics, and for standing as my role model in leadership. I thank Osbert Bastani for always meeting with me anytime I need and taking time to think through my ideas, and for polishing my paper and poster. I thank Jianbo Shi, Paul Yushkevich, and Elena Bernadis for guiding me into computer vision in my early research career.

I have been blessed with several friends at UPenn—Shuo Li, Zheyuan Xie, Dingding Zhen, Ruo Jia, Ziyun Wang, Xu Liu, Xiaoyi Chen, Yulai Weng, Feifei Duan, Jiahui Li, Shumin Yuan, Jiani Huang, Anton Xue, Siyao Hu, Lingzhi Zhang, Yinshuang Xu, Haimin Hu, Shuzhan Yang, Yufu Wang, Jiayue Wu, Yansong Gao, Rui Li, Mengjing Dong, Jingqi Li, Chenhao Liu—who have helped my grow and make my life much more fun. I thank all my collaborators and colleagues—Bernadette Bucher, Karl Schmeckpeper, Alex Zhu, Stephen Phillips, Monroe Kennedy III, Jimmy Paulos, Steven Chen, Wenxin Liu, Clark Zhang, Caio Mucchiani, Kishor Jothimurugan, Rahul Ramesh—who have provided me with helpful advice and make my Master's much more productive.

No words can express my gratitude to my parents, Chuankang Zhang and Xiuqin Chen, who are always the strong supporters behind me. Their unconditional love gives me the power to overcome the challenges in my life. I am very fortunate to be their son.

Abstract

Deformable objects manipulation, such as rope manipulation, is a challenging problem due to high-dimensionality, non-linearity, and hyper-underactuated systems. In this thesis, we first introduce some useful methods for image feature extraction. Then we elaborate on the idea of the Koopman operator to map a finite non-linear dynamical system into a linear dynamical system. We also propose a framework based on the Koopman operator and use it to predict the future rope states. Experimental results show our framework can be used to predict the rope states in short time horizons

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Related Work 3

1.3 Organization 4

2 Preliminary **6**

2.1 Problem Formulation 6

2.2 Notation 7

3 Image Feature Extraction **9**

3.1 Dataset 10

3.2 Color Space Segmentation 11

3.3 Active Contour Model 11

3.4 Cubic Spline 14

4 Model Reduction **15**

4.1 Properties 15

4.2 Model Reduction of Rope 16

5 System Identification **17**

5.1 Koopman Operators 18

5.2 External Control Input 19

5.3 Encoding Action 20

5.4 Learning Observation Function 20

5.5 Related Work 23

6 Deformable Linear Objects Manipulation **24**

6.1 Pipeline 24

6.2 Loss Function 27

6.3 Prediction Framework 27

6.4 Prediction Results 28

7 Concluding Remarks **30**

7.1 Contributions 30

7.2 Future Work 31

A Appendix A **32**

A.1 Code 32

A.2 Video 32

Bibliography **33**

List of Tables

2.1 Summary of the notation 8

List of Figures

1-1 Common manipulation tasks regarding deformable objects. Left to right, up to bottom: house-keeping tasks, cable manufacturing, surgery, and food-making. 2

3-1 Rope image before and after action 10

3-2 A sequence of procedures on color space segmentation. (a) Original image, (b) Original image + green mask, (c) Binary mask 11

3-3 Rope splines after applying Snakes method. Gray color represents the ground truth of the rope. Red dash lines show the automatically generated control points on the rope. Blue color denotes the fitted deformable splines. 14

4-1 Reduced model (right) [39] for a humanoid robot (left) 16

6-1 Model's pipeline one 25

6-2 Model's pipeline two 26

6-3 Model's pipeline three 26

6-4 Prediction framework 28

6-5 Prediction results 29

1 Introduction

> *A journey of a thousand miles must begin with a single step.*

Lao Tzu, *Dao De Jing*

Robotic manipulation can be applied to a variety of scenarios, such as surgery, housekeeping, manufacturing, and agriculture[45][22]. Deformable objects manipulation is one of the important robotic manipulation problems. We face deformable objects almost every day in our daily life (Fig. 1-1). e.g., folding the clothes[7], knot-typing[20], and moving a cable [44]. However, deformable object manipulation is still a challenging task due to 1) high model uncertainty, 2) high cost of simulation, 3) infinite-dimensional configuration spaces, and 4) hyper-underactuated systems.

1.1 Motivation

Since the dynamic for the deformable objects (e.g., rope) is non-linear, it's challenging to know this dynamic accurately. Model reduction is a good way to reduce the complexity of the system. e.g., [39] use cart and pole to simplify the one-legged model for humanoid

Figure 1-1: Common manipulation tasks regarding deformable objects. Left to right, up to bottom: house-keeping tasks, cable manufacturing, surgery, and food-making.

balance and gait analysis. Borrowing from the idea of model reduction, we can build a model that should capture the key representations for planning and control. The reduced model should follow four important properties: 1) easier to infer from sensor data, 2) faster to use in a planner loop, 3) less sensitive and more robust to error in parameters, and 4) more stable than physics simulation. Koopman operator [24][25] is a good method to map a finite non-linear dynamical system into an infinite linear dynamical system. There are also lots of good properties in linear dynamics, such as efficient estimation, control, and prediction. Thus, we use the Koopman operator to reduce the complexity of the dynamics and do planning in the linear space.

1.2 Related Work

There are a lot of research works trying to solve this problem from different perspectives:

State Estimation: Yan et al.[55] apply self-supervised learning for state estimation; Schulman et al.[47] incorporate the point cloud in the probabilistic generative model for tracking deformable objects; Tang et al.[50] introduce a tangent space mapping algorithm to maintain structural information when applying non-rigid registration; Wu et al.[54] use a convolutional neural network to find the physical object's state, which contains both intrinsic and extrinsic properties; Ehrhardt et al. [8]introduce two unsupervised principles of causality and equivariance and use them to track objects in the video without auto-encoding; Byravan et al.[5] introduce a structured deep dynamics model to learn a low-level pose embedding for control; Petit et al.[41] track objects with point cloud data from an RGB-D camera; Doumanoglou et al. [7] extract features from a depth map to select the grasping point.

Planning and Control: Yan et al.[55] use Model Predictive Path Integral (MPPI) controller for choosing the best grasping point with the least moving length to the goal. Moll et al.[35] compute the minimal-energy curves for path planning in a wire manipulation task. Roussel et al.[42] plan by contacts with extensible elastic rods. Bayazit et al.[3] introduce a Probabilistic Roadmap (PRM) method. There are also some other sampling-based methods which have been introduced.[2][12][28][35][42][44].

Deformable Objects Modeling: There are also a few methods modeling deformable objects. e.g., using Mass-Spring models[14][10], Finite Element models[41][36][16][19]. These approaches are useful for computer graphics with applications in movies and animations. However, it's challenging to learn all the model parameters accurately for deformable objects in the real world, and it takes a lot of time to simulate if we want to deploy the model on the robotic systems. Li et al.[30] build a database for 3D deformable objects retrieval.

It's efficient but we need to build these models with various poses beforehand.

Visual Servoing: Smolen et al.[48] propose the control based on a transformation between a set of manipulated point displacements and the control point displacement. Wada et al.[51] use PID feedback control to decrease the error between the model and the manipulated object. Navarro-Alarcon et al.[38] propose using feedback from the representation of the object's shape to help soft object manipulation. This representation is based on a truncated Fourier series. Navarro-Alarcon et al.[] introduce an adaptive method to servo-control deformations without calibrations for kinematic controllers. Other methods in this field have also been proposed. [4][33].

Imitation Learning: Nair et al. [37] use the human demonstration to teach the robot how to move the rope. The robot will learn the policy from the video sequence and apply this policy for execution. Huang et al.[15] approach the problem by trajectory transfer through non-rigid registration.

1.3 Organization

This thesis is organized as followed. In Chapter 1, we introduce the robotic manipulation tasks *in the wild* and the popularity of robotic manipulation in homes, factories, and hospitals. To be more specific, we introduce the problem of manipulating deformable objects and the reason why it's still challenging to solve. We also introduce some current works on deformable objects manipulation from various perspectives.

Chapter 2 formulates the problem of deformable object manipulation. We also introduce the notation.

In Chapter 3, we cover the methods to extract features from the images, including color space segmentation, an active contour model, and cubic spline. We also introduce how to use these features for state estimation.

Chapter 4 uses a model reduction example to illustrate the concept of the reduced model and its good properties.

Chapter 5 introduces the idea of the Koopman operator, the architecture of autoencoder, and how to use learning to identify the system in the linear latent space.

Chapter 6 builds the prediction framework based on both the idea of Koopman theory and autoencoders. We also show the experimental results and analysis based on the prediction output.

Finally, we take a holistic view of deformable objects manipulation using the Koopman operator. The last chapter also introduces the contributions and future directions for research.

2 Preliminary

> *From a teleological point of view the particular numerical solution of any particular set of equations is of far less importance than the understanding of the nature of the solution.*

Richard Bellman, *Dynamic Programming*, 1957

In this chapter, we first formulate the mathematical problem of rope manipulation in 2.1. Then we introduce the notations in 2.2. These notations will be used in the following Chapters.

2.1 Problem Formulation

We formulate the problem of deformable object (rope) manipulation as followed. Consider a discrete-time system with state $\mathbf{x}^t \in \mathcal{X} \subset \mathbb{R}^n$, control input $\mathbf{u}^t \in \mathcal{U} \subset \mathbb{R}^p$, observation $\mathbf{o}^t \in \mathcal{O} \subset \mathbb{R}^q$, a dynamical model $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$, and an observation function

$h : \mathbb{R}^n \rightarrow \mathbb{R}^q$ at time $t \in [T] = \{1, 2, ..., T\}$, we have

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t) \tag{2.1}$$

and

$$\mathbf{o}^t = h(\mathbf{x}^t) \tag{2.2}$$

Given an observation $\mathbf{o}^t$ (e.g., an RGB image) at time step $t = t'$, we want to learn its associated features $\mathbf{p}^t$ and try to predict the features at next time steps when $t = t' + 1, t' + 2, ..., T$.

2.2 Notation

Before introducing the technical details, we first summarize the notation in Table 2.1.

Table 2.1: Summary of the notation

symbol	definition	example/details
$\mathbf{x}^t$	system state at time step $t \in [T] = \{1, 2, ..., T\}$, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$	high-dimension rope state; the state includes a lot of physical properties: rope positions, stiffness, materials, etc.
$\mathbf{o}^t$	observation at time step $t \in [T]$, $\mathbf{o}^t \in \mathcal{O} \subset \mathbb{R}^q$	rope image from RGB camera
$\mathbf{u}^t$	control signal or system action at time step $t \in [T]$, $\mathbf{u}^t \in \mathcal{U} \subset \mathbb{R}^p$	joint motor torque commands; e.g., action includes grasping position (x, y) and moving distance (dx, dy)
$\mathbf{a}^t$	embedded action at time step $t \in [T]$, $\mathbf{a}^t \in \mathcal{A} \subset \mathbb{R}^k$	higher dimensional action space than the control input
$\mathbf{p}^t$	features of an observation $\mathbf{o}^t$	it can be points positions (x, y) in 2D space; or a spline; or a binary mask which is the segmentation of rope on the original image
$\mathbf{g}^t$	system embedded state at time step $t \in [T]$, $\mathbf{g}^t = g(\mathbf{x}^t) \subset \mathbb{R}^m$	embedding function can be a neural network; embedding function $g : \mathbb{R}^n \to \mathbb{R}^m$
$\mathbf{K}$	Koopman matrix $\mathbf{K} = \begin{bmatrix} k_{11} & \cdots & k_{1m} \\ \vdots & \ddots & \vdots \\ k_{m1} & \cdots & k_{mm} \end{bmatrix}_{m \times m} = \mathbf{I}_{m \times m}$	When there is no system action, system embedded state is unchanged: $\mathbf{g}^{t+1} = \mathbf{K}\mathbf{g}^t + \mathbf{L}\mathbf{u}^t$, $\mathbf{u}^t = \mathbf{0}$, and $\mathbf{g}^{t+1} = \mathbf{g}^t$. Therefore, Koopman matrix is identity matrix $\mathbf{I}$. We have a weak assumption that the robot manipulates the rope slowly so that the embedded state would be the same when there is no action.
$\mathbf{L}$	control matrix $\mathbf{L} = \begin{bmatrix} l_{11} & \cdots & l_{1p} \\ \vdots & \ddots & \vdots \\ l_{m1} & \cdots & l_{mp} \end{bmatrix}_{m \times p}$	control matrix can be learned by gradient backpropagation or learn it as the output of neural network

3 Image Feature Extraction

> *I am one of those who think like Nobel, that humanity will draw more good than evil from new discoveries.*
>
> Marie Curie, *Quotes*, Unknown

There are typically two kinds of works to learn a rope dynamics model: 1) learning dynamics directly in the image space or latent space [37], and 2) learning dynamics after explicit state estimation, e.g. object pose estimation [55]. Learning from images is widely used in the end-to-end learning pipeline. This method can provide rich visual information about the object state and it doesn't need much prior knowledge about the physics model. But many computation resources would be used to learn an accurate dynamics model. State estimation is to find the current state of a system, e.g., a quadrotor, a robot manipulator. Different from end-to-end learning, a lot of researchers separate the system into small components, such as the dynamics model, state estimation, and prediction model. State estimation incorporates the prior knowledge and it can largely reduce the noise in the image space. In this chapter, we will introduce several ways for image feature extraction to have a better state estimation.

3.1 Dataset

We are using the Berkeley's rope manipulation dataset[37]. There are more than sixty runs in this dataset. In each run, there are about 2000 images and their associated actions. Each action is

$$\mathbf{u} = [p_x, p_y, \theta, l, g], \tag{3.1}$$

where p_x and p_y are x, y-coordinates of the action in pixels, θ is the angle of action in radians, $l \in [0.01, 0.15]$ is the length of the action in meters, and $g \in \{0, 1\}$ indicates whether the action should be used for training. In Figure 3-1, the left image shows the current rope shape and black arrow is the action, the right image shows the rope shape after taking this action.

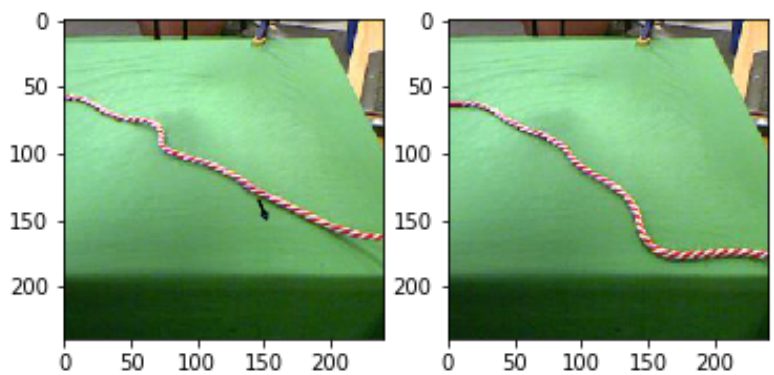

Figure 3-1: Rope image before and after action

3.2 Color Space Segmentation

Some works [37] use the raw images as the input in the training process. Here, we adopt a different strategy to segment the rope out of the image. Color spaces can represent the image color channels which gives the image certain hue. There are several popular color spaces, including *RGB* (Red, Green, Blue), *HSV* (Hue, Saturation, Value), and *CMYK* (Cyan, Magenta, Yellow, Black), etc. For rope images in Figure 3-2, we first use a green mask to cover the background color which is not green. Then we use *HSV* in the range for dark and light green to segment the green background. This method can easily provide us with the rope pose and shape for further analysis. It's also robust to different rope and background as long as there is a contrast between object color and background color.

(a) (b) (c)

Figure 3-2: A sequence of procedures on color space segmentation. (a) Original image, (b) Original image + green mask, (c) Binary mask

3.3 Active Contour Model

The active contour model is also named snakes, which is a method to find an object outline in a 2D image. This method is introduced by Michael Kass, Andrew Witkin, and Demetri Terzopoulos [18], and it's widely used on edge detection, object tracking, and shape recognition, etc.

Given constraints, image forces, and internal forces, a snake is matched to the object contours in the form of energy minimization. Consider a deformable spline defined by a set of N points $\mathbf{v}_i$, $i \in [N]$, we denote the internal energy term as E_{internal} and the external edge-based energy term as E_{external}.

The internal energy is the combination of the continuity of the contour E_{cont} and the smoothness of the contour E_{curv}. We can expand the internal energy into

$$\begin{aligned} E_{\text{internal}} &= E_{\text{cont}} + E_{\text{curv}} \\ &= \frac{1}{2}(\alpha(s)|\mathbf{v}_s(s)|^2) + \frac{1}{2}(\beta(s)|\mathbf{v}_{ss}(s)|^2) \\ &= \frac{1}{2}\Big(\alpha(s)||\frac{d\bar{v}}{ds}(s)||^2 + \beta(s)||\frac{d^2\bar{v}}{ds^2}(s)||^2\Big), \end{aligned}$$

where $\alpha(s)$ and $\beta(s)$ are user-defined weights.

The external energy is the sum of image energy E_{image} and the user-defined constraint forces E_{con}. The image energy is based on the line, edge, and termination features in the image. Its general form is

$$E_{\text{image}} = w_{\text{line}}E_{\text{line}} + w_{\text{edge}}E_{\text{edge}} + w_{\text{term}}E_{\text{term}},$$

where w_{line}, w_{edge}, and w_{term} are weights of these features. E_{line}, E_{edge}, and E_{term} are energy of lines, edges, and terminations. For an image $I(x, y)$, the line energy is

$$E_{\text{line}} = I(x, y),$$

or

$$E_{\text{line}} = \text{filter}(I(x, y))$$

if we use a filter. The edge energy is

$$E_{\text{edge}} = -|\nabla I(x, y)|^2$$

and it's based on the gradient of the image. Let $C(x, y)$ be the smoothed image after a Gaussian G_σ with standard deviation σ, then $C(x, y) = G_\sigma \cdot I(x, y)$ has gradient angle $\theta = \arctan(C_y/C_x)$. The unit vector perpendicular to the gradient direction is $\mathbf{n}_\perp = (-\sin\theta, \cos\theta)$. Then the termination energy is

$$\begin{aligned} E_{\text{term}} &= \frac{\partial\theta}{\partial\mathbf{n}_\perp} \\ &= \frac{C_{yy}C_x^2 - 2C_{xy}C_xC_y + C_{xx}C_y^2}{(C_x^2 + C_y^2)^{3/2}} \end{aligned}$$

The internal energy can control the snake deformations and the external energy can control the fitting of the contour onto the object. We add these terms above to the energy function of the snake in the form of

$$\begin{aligned} E^*_{\text{snake}} &= \int_0^1 E_{\text{snake}}(\mathbf{v}(s))ds \\ &= \int_0^1 (E_{\text{cont}}(\mathbf{v}(s)) + E_{\text{curv}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s)) + E_{\text{con}}(\mathbf{v}(s)))ds \end{aligned}$$

Before apply Snakes, we need to have sorted nodes according to the order of the rope. If the nodes are not in order, the spline will not match the rope shape. Consider a set of M nodes $\mathbf{m}_i$, $i \in [M]$, they are distributed on the rope. First, we build a graph $\mathcal{G}$ between these nodes based on 2-Nearest Neighbor, e.g., node $\mathbf{m}_i$ is connected with nodes $\mathbf{m}_j$ and $\mathbf{m}_k$. Let d_{ij} denotes *L2* distance between notes $\mathbf{m}_i$ and $\mathbf{m}_j$, and $d_{ij} = d_{ji}$. Define cost for

node $\mathbf{m}_i$ as

$$c_i = d_{ij} + d_{ik}.$$

In the outer loop, we pick an initial node with the lowest cost ; in the inner loop, we sum up the costs based on current initial node. The output is the node sequence based on the initial node with the lowest cost in graph $\mathcal{G}$.

After sorting the nodes and applying Snakes method to the binary rope mask in Fig. 3-2. we have the results in Fig. 3-3.

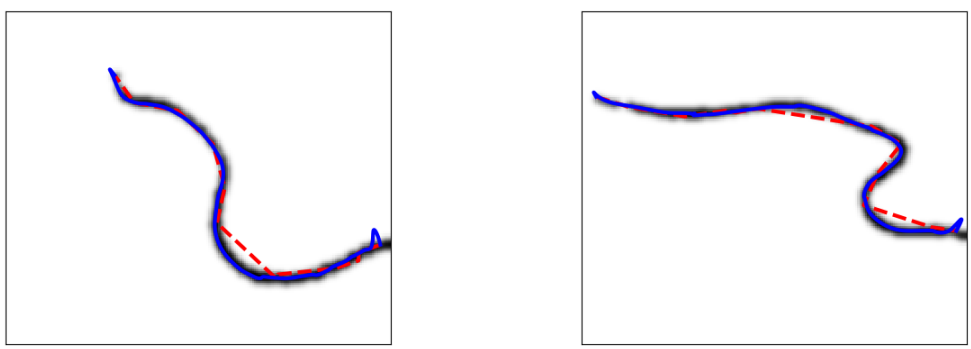

Figure 3-3: Rope splines after applying Snakes method. Gray color represents the ground truth of the rope. Red dash lines show the automatically generated control points on the rope. Blue color denotes the fitted deformable splines.

3.4 Cubic Spline

Fitting the control points with the cubic spline is another method to find the rope curve. When using this method, we assume the rope is composed of a piecewise cubic polynomial which is twice continuously differentiable. This assumption may not be true for real ropes. We test this method but find the spline is not perfectly aligned with the rope. We are not going to use this technique to find the rope spline.

4 Model Reduction

When solving the problem of manipulating deformable objects, researchers propose a lot of methods trying to model the deformable objects correctly, e.g., Mass-Spring models [14][10], Finite Element models[41][36][16][19]. These modeling methods are widely used in computer graphics, animations, and movies. However, for the deformable objects *in the wild*, it's challenging to find all correct model parameters for different kinds of objects. If a models is inaccurate, it will increase the difficulty of further planning, control, and prediction. In addition, it takes time to simulate the object. Thus, it's not applicable to use these methods in a real-world setting, e.g., a robot needs to react to the changes and manipulates the object in a real-time manner.

What can we do when the system is very complicated to model? Model reduction is a technique to reduce the model complexity while capturing the most important part of the model for planning and control. E.g., in Fig. 4-1 humanoid robots are hard to model and researchers would simply use a reduced one-legged model to analyze the humanoid balance and gait.

4.1 Properties

There are four important properties of a good reduced model:

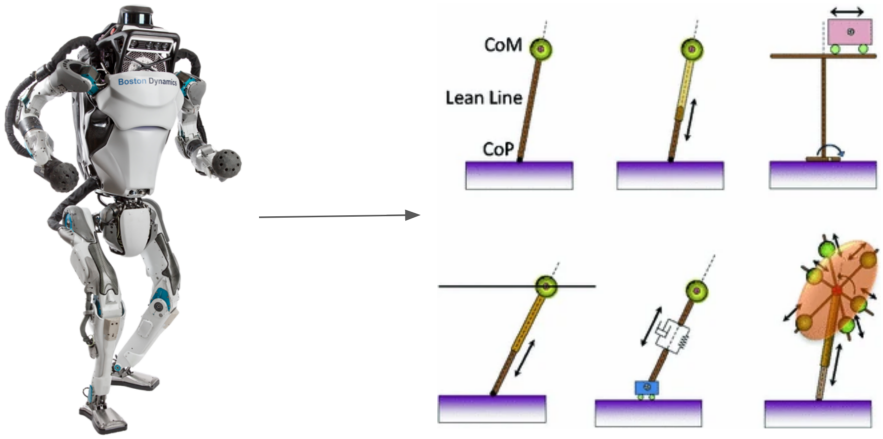

Figure 4-1: Reduced model (right) [39] for a humanoid robot (left)

Easier Inference The reduced model should be easier to infer from sensor data, compared with the original model.

Faster Planning This reduced model should be faster to use in a planner loop.

Less Sensitive This reduced model is robust to the error in parameters.

More stable The reduced model is more stable than physics simulation.

4.2 Model Reduction of Rope

Since the rope's dynamics are non-linear and its state space is high dimensional, we want to borrow from the idea of model reduction for humanoid robots and apply this idea to the rope manipulation. In the next Chapter 5, we will introduce the Koopman operator in detail and how to map the finite nonlinear dynamical system to the linear dynamical system based on the Koopman operator.

5 System Identification

> *One day, in retrospect, the years of struggle will strike you as the most beautiful.*
>
> Sigmund Freud, *Quotes*, Unknown

In applied mathematics and physics, the analysis of the nonlinear dynamical system has received a significant amount of attention because of its complex and nonlinear behaviors. A classical view of the dynamical system is to find the state-space models. However, the analysis becomes challenging when the dynamics are nonlinear and we cannot have an accurate state space model for this system.

There is an increasing interest to use operator-theoretic approaches to analyze the dynamical system, e.g., the Koopman operator. The Koopman operator first appears in the works of Koopman and Von Neumann in the early 1930s [24], [25]. The main idea is to map a finite-dimensional nonlinear dynamical system to an infinite-dimensional linear system based on the Koopman operator. Thus, we can analyze the evolution of observables in the linear function space rather than the state vectors in the nonlinear state space. This modal decomposition has been widely used in scientific domains and engineering prob-

lems. There are also several good properties to do analysis in the embedding space, e.g., we can have efficient estimation, control, and prediction using tools from linear dynamical systems. Recent works have successfully applied this method in soft robotics [31], model-based control[1], fluid mechanics[34], and image processing[9].

One popular modal decomposition algorithm based on the Koopman operator is Dynamic Mode Decomposition (DMD) [43][27][46]. An assumption to use DMD is that the observables are in a linear function space which is invariant to the Koopman operator. This space is also named Koopman invariant subspace. If we only have states in the nonlinear space, we should manually prepare observables based on the underlying nonlinear dynamics. People have been using hand-designed basis functions [53] or kernels [21] to solve this problem. However, these methods can only work well when we find appropriate basis functions or kernels, and finding a good basis function or kernel is heavily limited by our prior knowledge about the dynamics.

For a rope manipulation task, we can collect the data sequence by moving the rope and observing the new state of rope. Thus, we are going to use a data-driven method to identify the system in the Koopman invariant subspaces.

5.1 Koopman Operators

Let $\mathbf{x}^t \in \mathcal{X} \subset \mathbb{R}^n$ be the state vector for the system at time t. Consider a non-linear discrete-time dynamical system

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t), t \in [T] = \{0, 1, 2, ..., T\} \tag{5.1}$$

where $\mathcal{X}$ is the state space and $(\mathcal{X}, \Sigma, \mu)$ denotes the associated probability space. In dynamical system 5.1, *Koopman operator* $\mathcal{K}$ is defined as an infinite-dimensional linear op-

erator that applied on *observable* $g : \mathcal{X} \to \mathbb{R}^m$, i.e.,

$$\mathcal{K}g(\mathbf{x}^t) = g(f(\mathbf{x}^t)) = g(\mathbf{x}^{t+1}). \tag{5.2}$$

The analysis of nonlinear system 5.1 can be changed into an infinite-dimensional linear system in 5.2. Although the Koopman theory guarantees the existence of the Koopman operator, its practical application is limited by its infinite dimension. Thus, we slightly abuse the notation, and use a finite dimensional linear operator $\mathbf{K} \in \mathbb{R}^{m \times m}$ as the *Koopman operator*. In most cases, we assume there is a Koopman invariant subspace $\mathcal{G}$ based on Koopman operator. This invariant subspace is spanned by a set of observables $\{g_1, g_2, ..., g_m\}$, where $\mathcal{K}g \in \mathcal{G}$ for any $g \in \mathcal{G}$. For simplification, we use observation function $g(\mathbf{x}^t) : \mathbb{R}^n \to \mathbb{R}^m$ to represent $[g_1(\mathbf{x}^t), g_2(\mathbf{x}^t), ..., g_m(\mathbf{x}^t)]^T$. Rewrite 5.2 based on the the finite dimensional linear operator $\mathbf{K}$, we have

$$\mathbf{K}g(\mathbf{x}^t) = g(f(\mathbf{x}^t)) = g(\mathbf{x}^{t+1}). \tag{5.3}$$

5.2 External Control Input

Consider a system with state $\mathbf{x}^t \in \mathcal{X} \subset \mathbb{R}^n$, control input $\mathbf{u}^t \in \mathcal{U} \subset \mathbb{R}^p$, and a dynamical model $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$, we have

$$\mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t). \tag{5.4}$$

We are going to find the Koopman operator $\mathbf{K}$, control matrix $\mathbf{L}$, and Koopman observation function in the form of

$$g(\mathbf{x}^{t+1}) = \mathbf{K}g(\mathbf{x}^t) + \mathbf{L}\mathbf{u}^t \tag{5.5}$$

5.3 Encoding Action

Since the size of the control input $\mathbf{u}$ is small, we can also encode the control input to a larger size so that the action $\mathbf{a}$ in the latent space can have more physical meanings. Thus, we can modify Eq. 5.5 to

$$g(\mathbf{x}^{t+1}) = \mathbf{K}g(\mathbf{x}^t) + \mathbf{L}\mathbf{a}^t \tag{5.6}$$

Expanding this equation, we get

$$\underbrace{\begin{bmatrix} g_1^{t+1} \\ \vdots \\ g_m^{t+1} \end{bmatrix}}_{\mathbf{g}^{t+1}} = \underbrace{\begin{bmatrix} k_{11} & \cdots & k_{1m} \\ \vdots & \ddots & \vdots \\ k_{m1} & \cdots & k_{mm} \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} g_1^t \\ \vdots \\ g_m^t \end{bmatrix}}_{\mathbf{g}^t} + \underbrace{\begin{bmatrix} l_{11} & \cdots & l_{1p} \\ \vdots & \ddots & \vdots \\ l_{m1} & \cdots & l_{mp} \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} a_1^t \\ \vdots \\ a_p^t \end{bmatrix}}_{\mathbf{a}^t} \tag{5.7}$$

5.4 Learning Observation Function

The classical method is to manually design an observation function based on the knowledge of physics. Deep neural networks is good to learn the observation function $g(\mathbf{x})$ from data. One popular type of deep neural networks is an autoencoder.

The autoencoder can learn the representations by encoding the state $\mathbf{x}$ into embedding state $\mathbf{g} = \phi_\theta(\mathbf{x})$, where ϕ_θ is an encoding neural network parameterized by neural network weights θ. Another part of the autoencoder is a decoding neural network ϕ_α which is parameterized by weights α. It can decode the representations $\mathbf{g}$ into reconstructed state $\mathbf{x}$. We formulate the reconstruction loss to train the autoencoder to be

$$\sum_{t=0}^{T} ||\phi_\alpha(\phi_\theta(\mathbf{x}^t)) - \mathbf{x}^t||_2^2 \tag{5.8}$$

and we minimize the $L2$ loss over weights θ and α. Since the structure of encoding neural network is similar to observation function, we will use $\mathbf{g} = g(\mathbf{x}) = \phi_\theta(\mathbf{x})$ to unify the notation.

We also encode the control input $\mathbf{u}$ to latent action $\mathbf{a}$. The loss function is

$$\sum_{t=0}^{T-1} ||\phi_\gamma(\phi_\beta(\mathbf{u}^t)) - \mathbf{u}^t||_2^2, \tag{5.9}$$

where ϕ_β is the encoder, ϕ_γ is the decoder, and β, γ are the weights of neural networks.

Besides the loss of autoencoder, we set constraints 5.6 in the Koopman invariant subspace. We will minimize

$$\sum_{t=0}^{T} ||\mathbf{K}g(\mathbf{x}^t) + \mathbf{L}\mathbf{a}^t - g(\mathbf{x}^{t+1})||_2^2 \tag{5.10}$$

When there is no system action $\mathbf{u}$, we assume the rope state $\mathbf{x}$ is unchanged. Then 5.6 will be

$$g(\mathbf{x}^{t+1}) = \mathbf{K}g(\mathbf{x}^t) + \mathbf{L}\mathbf{a}^t = \mathbf{K}g(\mathbf{x}^t),$$

and Koopman matrix $\mathbf{K}$ is indentity matrix $\mathbf{I} \in \mathbb{R}^{m\times m}$. This assumption isn't strong because rope is static before and after each action. Thus, we only need to learn observation function g, control matrix $\mathbf{L}$, and weights of autoencoder for action in the form of

$$g(\mathbf{x}^{t+1}) = g(\mathbf{x}^t) + \mathbf{L}\mathbf{a}^t \tag{5.11}$$

$$= g(\mathbf{x}^t) + \mathbf{L}\phi_\gamma(\phi_\beta(\mathbf{u}^t)) \tag{5.12}$$

Since the dynamics are linear in the invariant space, we can write in recursion:

$$
\begin{aligned}
\mathbf{g}^1 &= \mathbf{g}^0 + \mathbf{L}\mathbf{a}^0 \\
\mathbf{g}^2 &= \mathbf{g}^1 + \mathbf{L}\mathbf{a}^1 = \mathbf{g}^0 + \mathbf{L}(\mathbf{u}^0 + \mathbf{a}^1) \\
&\vdots \\
\mathbf{g}^k &= \mathbf{g}^{k-1} + \mathbf{L}\mathbf{a}^{k-1} = \mathbf{g}^{k-1} + \mathbf{L}\sum_{t=0}^{k-1}\mathbf{a}^t.
\end{aligned}
$$

Combining the last equation in this recursion and 5.8, we have a new loss for autoencoder in k steps:

$$
\sum_{t=0}^{T-k} ||\phi_\alpha(\mathbf{g}^{t+k}) - \mathbf{x}^{t+k}||_2^2 = \sum_{t=0}^{T-k} ||\phi_\alpha(\mathbf{g}^t + \mathbf{L}\sum_{l=0}^{k-1}\mathbf{a}^{t+l}) - \mathbf{x}^{t+k}||_2^2. \tag{5.13}
$$

Finally, by adding 5.8, 5.9, 5.10, and 5.13, we will minimize the loss

$$
\begin{aligned}
\mathcal{L}(\mathbf{L}, \alpha, \theta, \beta, \gamma) = &\sum_{t=0}^{T} ||\phi_\theta(\mathbf{x}^t) + \mathbf{L}\phi_\beta(\mathbf{u}^t) - \phi_\theta(\mathbf{x}^{t+1})||_2^2 \\
&+ \lambda_1 \sum_{t=0}^{T} ||\phi_\alpha(\phi_\theta(\mathbf{x}^t)) - \mathbf{x}^t||_2^2 \\
&+ \lambda_2 \sum_{t=0}^{T-k} ||\phi_\alpha(\phi_\theta(\mathbf{x}^t) + \mathbf{L}\sum_{l=0}^{k-1}\phi_\beta(\mathbf{u}^{t+l})) - \mathbf{x}^{t+k}||_2^2 \\
&+ \lambda_3 \sum_{t=0}^{T-1} ||\phi_\gamma(\phi_\beta(\mathbf{u}^t)) - \mathbf{u}^t||_2^2,
\end{aligned}
$$

where λ_1, λ_2, and λ_3 are the regularization terms. An illustrative framework to formulate this loss function is shown in Fig. 6-1.

5.5 Related Work

There are active studies about learning a model for nonlinear dynamical systems. Ghahramani et al.[13] apply the classical Expectation-Maximization algorithm to solve the nonlinear dynamical system. Kingma et al.[23] introduce the variational autoencoder using approximate Bayesian inference. Krishnan et al.[26] build a structured inference network to model nonlinear state space. Watter et al.[52] linearize the dynamics locally in the latent space for control from unprocessed images. Li et al.[29] utilize dictionary learning to extend dynamic mode decomposition. Karl et al.[17] introduce deep variational Bayes filters to learning state space from raw data. Yeung et al. [56] use the deep neural networks to learn representations for the Koopman operator. Otto et al.[40] apply linearly-recurrent autoencoder to learn dynamics. Susuki et al[49] use prony analysis to approximate Koopman mode decomposition. There are also other related works using neural networks to solve dynamics problems. [32][11][6]

6 Deformable Linear Objects Manipulation

> *The whole history of science has been the gradual realization that events do not happen in an arbitrary manner, but that they reflect a certain underlying order, which may or may not be divinely inspired.*

Stephen Hawking, *A Brief History of Time*, 1988

6.1 Pipeline

The pipeline is shown in Fig. 6-1. For each column, the rope state is $\mathbf{x}$, observation $\mathbf{o}$ is the RGB image from the camera, $\mathbf{p}$ is the feature and it can be points on the rope, a spline, or a binary mask for rope, $\mathbf{g}$ is the embedded state, and $\hat{\mathbf{p}}$ is the reconstructed feature. As for the actions, $\mathbf{u}$ is the control input, $\mathbf{a}$ is the embedded action, and $\hat{\mathbf{u}}$ is the reconstructed control input. There are two rows in this pipeline: the top row shows the nonlinear dynamics for the rope, which is $\mathbf{x}^{t+1} = f(\mathbf{x}^t, \mathbf{u}^t)$. The other row in this pipeline shows the linear dynamics in the latent space, which is $\mathbf{g}^{t+1} = \mathbf{K}\mathbf{g}^t + \mathbf{L}\mathbf{a}^t$. There are three constraints: a linear dynamic constraint in the latent space, a constraint of autoencoder for control input reconstruction,

and a constraint of convolutional autoencoder for feature reconstruction. We can then use these three constraints to formulate the loss function to train the neural network.

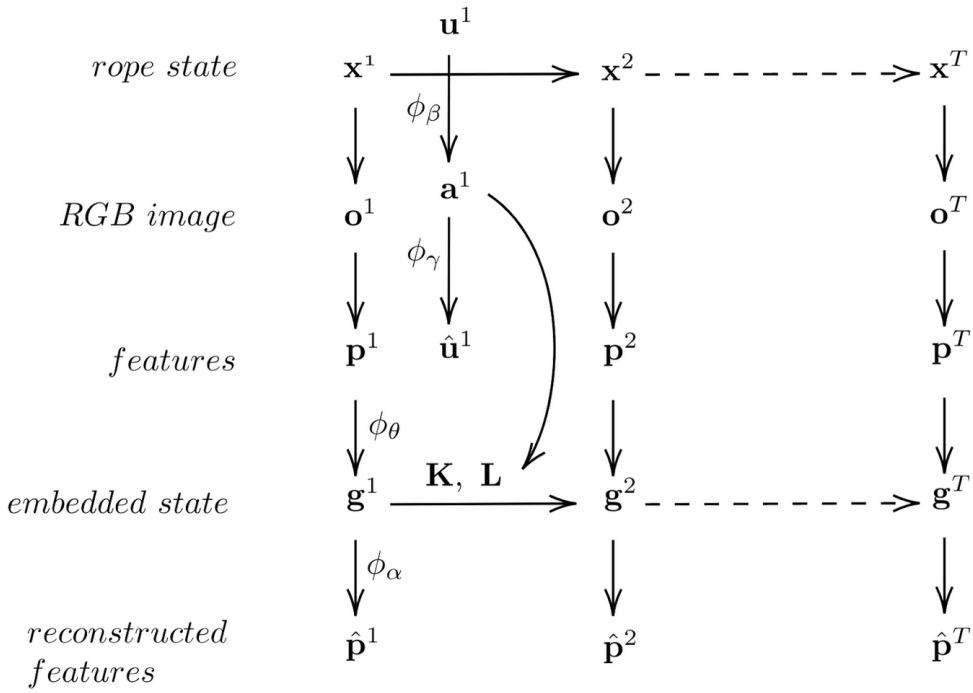

Figure 6-1: Model's pipeline one

We also design other pipelines shown in Fig. 6-2 and 6-3. In pipeline two, the main difference is that we directly use the real control input **u** as the action in the latent space. It will build a different manifold from the first pipeline's manifold. For pipeline three, we use multiple features (e.g. points position, spline, binary mask) as the inputs to the autoencoder.

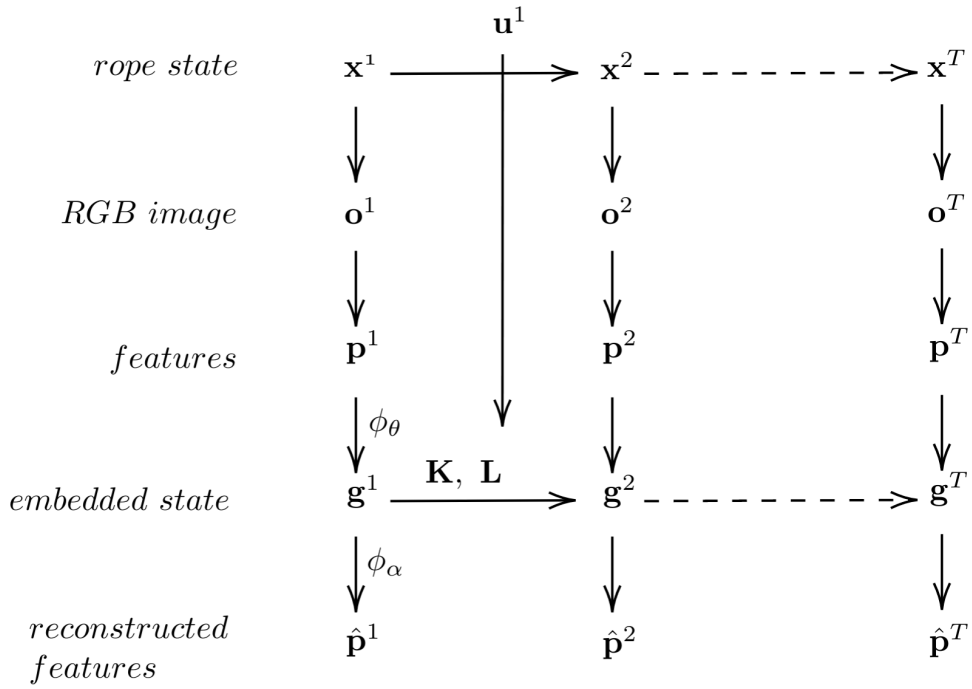

Figure 6-2: Model's pipeline two

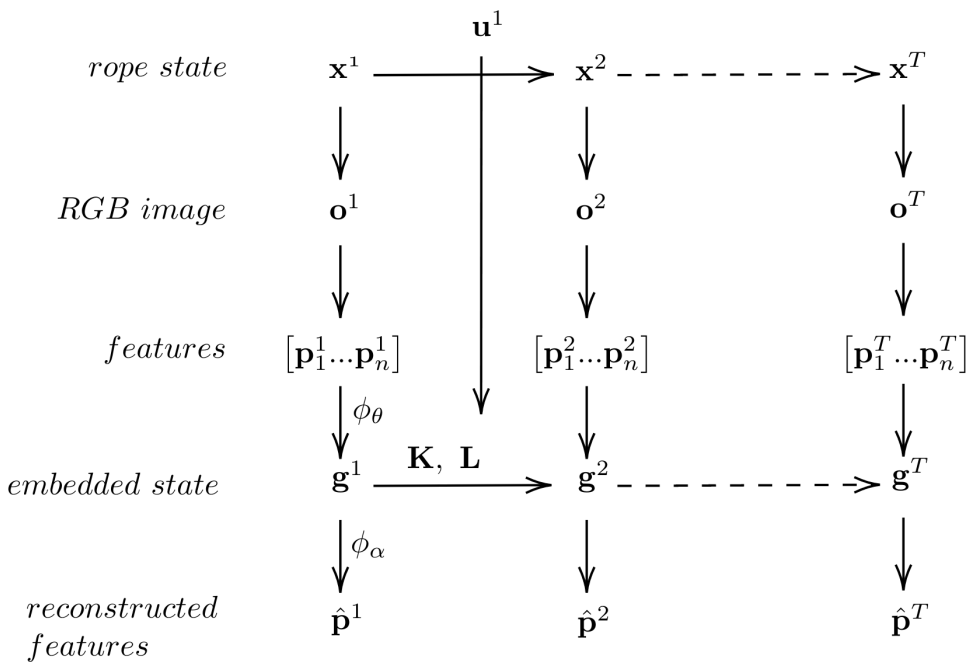

Figure 6-3: Model's pipeline three

6.2 Loss Function

Slightly different from the loss function described in Sec. 5, we set the features $\mathbf{p}$ to replace the state $\mathbf{x}$. Here, the first term is the constraint in latent space. The second term is the constraint for convolutional autoencoder. The third term is a constraint on k-step prediction loss, we can derive it by combining the first term and the second term. The fourth term is the constraint for autoencoder.

$$\begin{aligned}\mathcal{L}(\mathbf{L}, \alpha, \theta, \beta, \gamma) = &\sum_{t=0}^{T} ||\phi_\theta(\mathbf{p}^t) + \mathbf{L}\phi_\beta(\mathbf{u}^t) - \phi_\theta(\mathbf{p}^{t+1})||_2^2 \\ &+ \lambda_1 \sum_{t=0}^{T} ||\phi_\alpha(\phi_\theta(\mathbf{p}^t)) - \mathbf{p}^t||_2^2 \\ &+ \lambda_2 \sum_{t=0}^{T-k} ||\phi_\alpha(\phi_\theta(\mathbf{p}^t) + \mathbf{L}\sum_{l=0}^{k-1} \phi_\beta(\mathbf{u}^{t+l})) - \mathbf{p}^{t+k}||_2^2 \\ &+ \lambda_3 \sum_{t=0}^{T-1} ||\phi_\gamma(\phi_\beta(\mathbf{u}^t)) - \mathbf{u}^t||_2^2,\end{aligned}$$

where λ_1, λ_2, and λ_3 are the regularization terms.

6.3 Prediction Framework

The prediction framework is shown in Fig.6-4. Start from the left of this framework; the raw image is first processed by color space segmentation. Then the binary mask will be taken as an input to the convolutional autoencoder and the output is the embedding state vector. An associated action will also be encoded into an embedding action vector. By envolving the linear dynamics in the latent space, we can have the next embedding state based on Eq. 5.6. In the last step, the embedding state will be decoded into the predicted binary mask.

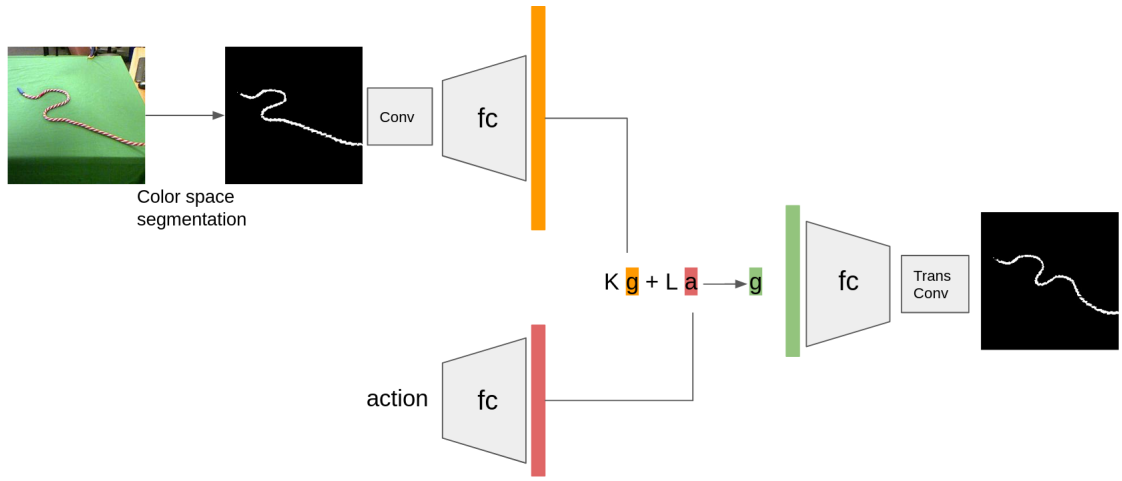

Figure 6-4: Prediction framework

6.4 Prediction Results

After training our neural network on Berkeley's rope dataset[37], we can have the values for the weights of neural networks, Koopman matrix, and control matrix. Based on that, we can further calculate the next embedding states in the latent space and decode it into the predicted images.

We input the initial image at time step $t = 0$ into our network to predict the future images. The results are in Fig. 6-5. The first row shows the ground truth and the images in the second row are our results. For time step $t = 1$, $t = 2$, and $t = 3$, we can find the rope's shape is similar to the ground truth while the rope loses some continuities. At the time step $t = 4$, the prediction starts to be distorted.

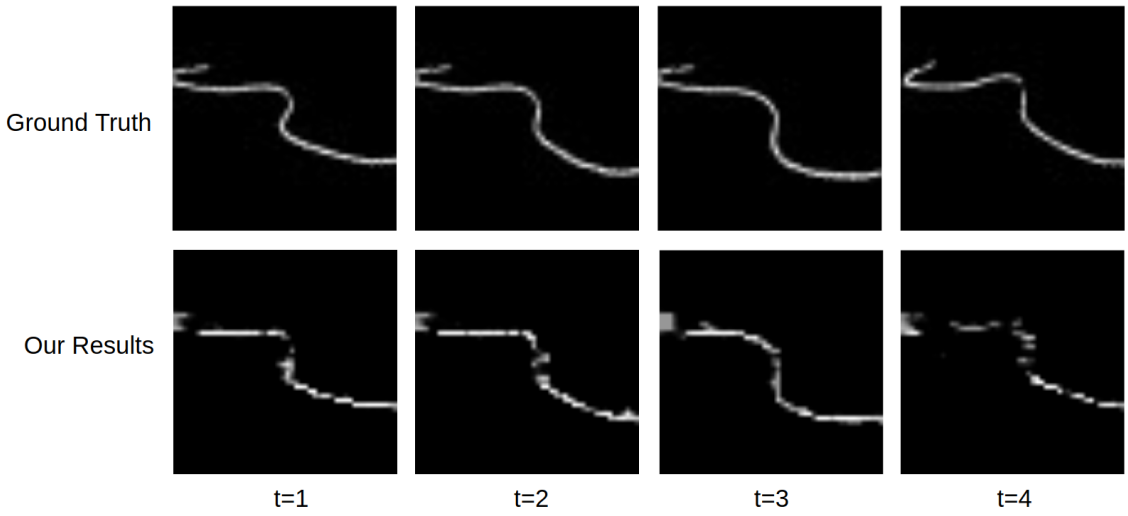

Figure 6-5: Prediction results

7 Concluding Remarks

> *The word "discovery" in itself is regrettable. For discovery is equivalent to becoming aware of a thing which is already formed; this links up with proof, which no longer bears the character of "discovery" but, in the final analysis, of the means that leads to discovery...Discovery is really not a creative act.*

Albert Einstein, *The Ultimate Quotable Einstein*, 2011

7.1 Contributions

The main contribution of this thesis is to build a new framework that incorporates the Koopman operator in the latent space for predicting future states of deformable objects. This framework can be used in the robotic manipulation tasks when we only have the observation but cannot get an accurate physics model of the object in real time. By applying the Koopman operator in the latent space, we can map the non-linear dynamical system into the linear dynamical system for efficient estimation, planning, and control. In the experiments, we test our method on a rope dataset. By comparing our predictions with

the ground truth images, we find the results show our method is robust for prediction in short-time horizons.

7.2 Future Work

Visual prediction based on the Koopman operator in this thesis is a novel way to solve the existing challenging planning and control problems from the perspective of computer vision. The framework in this thesis can also be used in several interesting and promising research directions.

Real Robotic Manipulation Our current results are based on the image predictions in a rope dataset. We can further explore this framework in the real-world robotics setting for rope manipulation.

Generalization We can also consider to improve the generalization ability of this framework. After training on a dataset, the weights of the neural network can be set as the initial weights to train new rope images.

Other Deformable Objects There are other types of deformable objects in our daily life, such as clothes, shoelaces, cables, etc. These objects share similar properties and future work can be built upon manipulating these objects.

Appendix A

A.1 Code

We publish our code on the GitHub written in Python: https://github.com/zwbgood6/deform

A.2 Video

We also have two videos which show a rope changing its shape from a line into a knot and a loop. Line to loop: https://www.youtube.com/watch?v=42YWPT3EXvM. Line to knot: https://www.youtube.com/watch?v=feJKQigyOzs.

Bibliography

[1] Ian Abraham, Gerardo De La Torre, and Todd D Murphey. Model-based control using koopman operators. *arXiv preprint arXiv:1709.01568*, 2017.

[2] Elliot Anshelevich, Scott Owens, Florent Lamiraux, and Lydia 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)*, volume 3, pages 2290–2295. IEEE, 2000.

[3] O Burchan Bayazit, Jyh-Ming Lien, and Nancy M Amato. Probabilistic roadmap motion planning for deformable objects. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 2126–2133. IEEE, 2002.

[4] Dmitry Berenson. Manipulation of deformable objects without modeling and simulating deformation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4525–4532. IEEE, 2013.

[5] Arunkumar Byravan, Felix Leeb, Franziska Meier, and Dieter Fox. Se3-pose-nets: Structured deep dynamics models for visuomotor planning and control. *arXiv preprint arXiv:1710.00489*, 2017.

[6] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[7] Andreas Doumanoglou, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimir Petrik, Andreas Kargakos, Libor Wagner, Václav Hlaváč, Tae-Kyun Kim, and Sotiris Malassiotis. Folding clothes autonomously: A complete pipeline. *IEEE Transactions on Robotics*, 32(6):1461–1478, 2016.

[8] Sebastien Ehrhardt, Aron Monszpart, Niloy Mitra, and Andrea Vedaldi. Unsupervised intuitive physics from visual observations. In *Asian Conference on Computer Vision*, pages 700–716. Springer, 2018.

[9] N Benjamin Erichson, Steven L Brunton, and J Nathan Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, 16(5):1479–1492, 2019.

[10] Nabil Essahbi, Belhassen Chedli Bouzgarrou, and Grigore Gogu. Soft material modeling for robotic manipulation. In *Applied Mechanics and Materials*, volume 162, pages 184–193. Trans Tech Publ, 2012.

[11] Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. In *Advances in neural information processing systems*, pages 163–171, 2016.

[12] Russell Gayle, Ming C Lin, and Dinesh Manocha. Constraint-based motion planning of deformable robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1046–1053. IEEE, 2005.

[13] Zoubin Ghahramani and Sam T Roweis. Learning nonlinear dynamical systems using an em algorithm. In *Advances in neural information processing systems*, pages 431–437, 1999.

[14] Sarah FF Gibson and Brian Mirtich. A survey of deformable modeling in computer graphics. Technical report, Citeseer, 1997.

[15] Sandy H Huang, Jia Pan, George Mulcaire, and Pieter 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)*, pages 878–885. IEEE, 2015.

[16] Geoffrey Irving, Joseph Teran, and Ronald Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 131–140, 2004.

[17] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

[18] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[19] Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. Flexible simulation of deformable models using discontinuous galerkin fem. *Graphical Models*, 71(4):153–167, 2009.

[20] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[21] Yoshinobu Kawahara. Dynamic mode decomposition with reproducing kernels for koopman spectral analysis. In *Advances in neural information processing systems*, pages 911–919, 2016.

[22] Fouad F Khalil and Pierre Payeur. Dexterous robotic manipulation of deformable objects with multi-sensory feedback-a review. In *Robot Manipulators Trends and Development*. IntechOpen, 2010.

[23] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, 2014.

[24] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

[25] BO Koopman and J v Neumann. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences of the United States of America*, 18(3):255, 1932.

[26] Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Thirty-first aaai conference on artificial intelligence*, 2017.

[27] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[28] Florent Lamiraux and Lydia E Kavraki. Planning paths for elastic objects under manipulation constraints. *The International Journal of Robotics Research*, 20(3):188–208, 2001.

[29] Qianxiao Li, Felix Dietrich, Erik M Bollt, and Ioannis G Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[30] Yinxiao Li, Yan Wang, Yonghao Yue, Danfei Xu, Michael Case, Shih-Fu Chang, Eitan Grinspun, and Peter K Allen. Model-driven feedforward prediction for manipulation of deformable objects. *IEEE Transactions on Automation Science and Engineering*, 15(4):1621–1638, 2018.

[31] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

[32] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[33] Dale McConachie and Dmitry Berenson. Estimating model utility for deformable object manipulation using multiarmed bandit methods. *IEEE Transactions on Automation Science and Engineering*, 15(3):967–979, 2018.

[34] Igor Mezić. Analysis of fluid flows via spectral properties of the koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013.

[35] Mark Moll and Lydia E Kavraki. Path planning for deformable linear objects. *IEEE Transactions on Robotics*, 22(4):625–636, 2006.

[36] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 49–54, 2002.

[37] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2146–2153. IEEE, 2017.

[38] David Navarro-Alarcon and Yun-Hui Liu. Fourier-based shape servoing: a new feedback method to actively deform soft objects into desired 2-d image contours. *IEEE Transactions on Robotics*, 34(1):272–279, 2017.

[39] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous robots*, 35(2-3):161–176, 2013.

[40] Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

[41] Antoine Petit, Vincenzo Lippiello, and Bruno Siciliano. Real-time tracking of 3d elastic objects with an rgb-d sensor. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3914–3921. IEEE.

[42] Olivier Roussel, Andy Borum, Michel Taix, and Timothy 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)*, pages 3116–3121. IEEE, 2015.

[43] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[44] Mitul Saha and Pekka Isto. Motion planning for robotic manipulation of deformable linear objects. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2478–2484. IEEE, 2006.

[45] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37(7):688–716, 2018.

[46] Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[47] John Schulman, Alex Lee, Jonathan Ho, and Pieter Abbeel. Tracking deformable objects with point clouds. In *2013 IEEE International Conference on Robotics and Automation*, pages 1130–1137. IEEE, 2013.

[48] Jerzy Smolen and Alexandru Patriciu. Deformation planning for robotic soft tissue manipulation. In *2009 Second International Conferences on Advances in Computer-Human Interactions*, pages 199–204. IEEE, 2009.

[49] Yoshihiko Susuki and Igor Mezić. A prony approximation of koopman mode decomposition. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 7022–7027. IEEE, 2015.

[50] Te Tang, Changliu Liu, Wenjie Chen, and Masayoshi Tomizuka. Robotic manipulation of deformable objects by tangent space mapping and non-rigid registration. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2689–2696. IEEE, 2016.

[51] Takahiro Wada, Shinichi Hirai, Sadao Kawamura, and Norimasa Kamiji. Robust manipulation of deformable objects by a simple pid feedback. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 85–90. IEEE, 2001.

[52] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

[53] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[54] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, pages 153–164, 2017.

[55] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. Self-supervised learning of state estimation for manipulating deformable linear objects. *IEEE Robotics and Automation Letters*, 5(2):2372–2379, 2020.

[56] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.