NAIST-IS-DD1361008

Doctoral Dissertation

Manifold Learning from High-Dimensional Data for System Modeling, Prediction and Robot Tactile Perception

Daisuke Tanaka

March 14, 2016

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Daisuke Tanaka

Thesis Committee:

Professor Kenji Sugimoto	(Supervisor)
Professor Tsukasa Ogasawara	(Co-supervisor)
Associate Professor Takamitsu Matsubara	(Co-supervisor)
Assistant Professor Yuki Minami	(Co-supervisor)
Professor Kentaro Hirata	(Okayama University)

Manifold Learning from High-Dimensional Data for System Modeling, Prediction and Robot Tactile Perception*

Daisuke Tanaka

Abstract

Recently, systems that employ various sensors have been constructed, and thus the amount of the sensor data in electronic form is growing. The combined data could be high dimensional, and modeling from this data may output poor models since it is difficult to fill the high-dimensional space. If we use the model constructed with high dimensional data, the prediction of the model with new data may be irrelevant. That is, the curse of dimensionality could be a problem. Meanwhile, since the high-dimensional data often lies on a lower-dimensional manifold intrinsically, the low-dimensional manifold can be extracted by applying the manifold learning techniques, also known as nonlinear dimensionality reduction methods. Then, we provide a better model, thus avoiding the curse of dimensionality. However, the model constructed with the low-dimensional data obtained with such a method may deteriorate the task performance or can lead to task failure since the method does not consider the task.

Thus, in this dissertation, we propose task-relevant manifold learning methods through construction of new criteria by adding constraints induced from the task such as prediction or control. By introducing this criterion, we achieve both manifold learning and modeling simultaneously, and the model is constructed that ameliorates the task performance. The proposition is validated through the following two problems.

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1361008, March 14, 2016.

First, we consider the linear system identification method using high dimensional input and output data. We assume that the input and output data lie on manifolds respectively, and each manifold relates dynamically. In this study, we propose a new criterion to obtain a model for accurate prediction by considering the error of fitting to the linear dynamics. Using this criterion, we achieve both the dimensionality reduction of the input and output data, and the system identification. We experimentally validate the effectiveness of the proposed method by experimenting with synthetic data.

Next, a partial manifold learning method is considered. We consider a two-factor generative model whose one factor is known and the other is unknown. In order to obtain an unknown factor considering the effect of both factors to the observation, we propose a manifold learning method that uses a prior information of the model structure as the constraints for optimization of the criteria. The proposed method is used for modeling for the object recognition problem using a robot hand with tactile sensors. The object recognition is achieved by estimating the inherent parameter of objects allocated in advance, by executing the exploratory action by the robot to the object. In order to achieve the recognition efficiently, we need a sensor model, which represents the relationship among the sensor data, the exploratory action, and the object being touched, to determine the informativeness of the resulting sensor data of the action. It is better to use the smooth model to make the recognition easier. Thus, we apply the proposed method in order to estimate the parameter so that a smooth model is obtained. The effectiveness of the proposed methods is validated through the numerical simulation and the experiments with the actual robot.

Keywords:

Manifold Learning, System Identification, Optimal Control, Isomap, Gaussian Process Latent Variable Models, Multi-Eigenvalue Problem

システムモデリング・予測・ロボットの触知覚のための高次元データからの多様体学習*

田中 大介

内容梗概

近年、多様なセンサにより構成されたシステムが構築されている。予測や制御を目的として、システムから得られるデータからモデリングを行う際、センサから得られる全てのデータをまとめると高次元になり、次元の呪いが問題になる。一方、高次元データは本質的には低次元の多様体上に偏って存在していることが多いため、これを抽出する非線形次元削減法である多様体学習法を適用することで、次元の呪いを回避しながらモデリングを達成できる。しかし、このように得た低次元データから得たモデルによりタスクを実行した場合には、タスクのことを考慮した次元削減が行われていないため、タスクが成功しない場合がある。

そこで本論文では、従来多様体学習法で用いられる評価基準に、予測や制御などの目的に起因する制約を加え、新たな評価基準を構築することでタスクに適した多様体学習法の構築を試みる。また、これにより多様体学習とモデリングを同時に達成し、タスク達成度を高める。この提案法の検証を以下の 2 つの課題を通して行う。

まず、高次元入出力データから線形システムを同定する手法を検討する。ここでは得られた入出力データがそれぞれある多様体上に拘束され、また多様体間には線形システムで表される動的な関係が存在すると仮定する。本研究では予測精度の高いモデルを得るため、低次元表現間の線形ダイナミクスへの当てはめ誤差を同時に考慮する新たな評価基準を提案する。この評価基準を用いることで、入出力データの次元削減と線形システムの同定を同時に達成する。提案手法の有効性は人工データを用いて実験的に検証し確認した。

*奈良先端科学技術大学院大学 情報科学研究科 博士論文, NAIST-IS-DD1361008, 2016 年 3 月 14 日.

次に、2因子生成モデルに対する部分多様体学習を考える。ここでは2因子のうち1つが既知でもう1つが未知の因子の生成モデルを考える。未知の因子を求めるために、本研究では、多様体学習の低次元表現を求める最適化問題に対して、事前に得られているモデル構造を制約条件として加える事で未知の因子を求める手法を提案する。本研究では、提案手法を触覚センサ付きロボットハンドによる物体認識問題に対するモデリングに対して適用する。物体認識は、各物体に予め割当てられた固有のパラメータを、探索行動により得られるセンサデータから推定することで達成する。効率的な認識を達成するためには、センサデータの有益さを測るためのセンサデータと行動と物体との関係を表すセンサモデルが必要となるが、推定問題を簡単にするためには、モデルが滑らかであることが必要である。本研究では、提案手法を適用することで、滑らかなモデルを得られるような各物体のパラメータ割当を達成する。提案手法は人工データと実機ロボットによる検証により有効性を確認した。

キーワード

多様体学習, システム同定, 最適制御, 等長特徴写像, ガウス過程潜在変数モデル, 多変量固有値問題

List of Publications

Journal Publications

1. 田中大介, 松原崇充, 杉本謙二, “入出力多様体学習によるシステム同定”, 電子情報通信学会論文誌 A, Vol. J96-A, No. 8, pp. 551–561, 2013.
   [Corresponds to Chapter 3]

2. Daisuke Tanaka, Takamitsu Matsubara and Kenji Sugimoto, “Input-Output Manifold Learning with State Space Models”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E99-A, No. 6, 2016. (to appear)
   [Corresponds to Chapter 4]

International Conference Proceedings

1. Daisuke Tanaka, Takamitsu Matsubara, Kentaro Ichien and Kenji Sugimoto, “Object Manifold Learning with Action Features for Active Tactile Object Recognition”, 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2014), MoB2.14, pp. 608–614, Chicago, Illinois, USA, September 14–18, 2014.
   [Corresponds to Chapter 5]

2. Daisuke Tanaka, Takamitsu Matsubara and Kenji Sugimoto, “An Optimal Control Approach for Exploratory Actions in Active Tactile Object Recognition”, 2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids2014), pp. 787–793, Madrid, Spain, November 18–20,

2014.
[Corresponds to Chapter 5]

3. Takamitsu Matsubara, Jaime Valls Miró, Daisuke Tanaka, James Poon and Kenji Sugimoto, “Sequential Intention Estimation of a Mobility Aid User for Intelligent Navigational Assistance”, 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 444–449, Kobe, Hyogo, Japan, August 31–September 4, 2015.
[Corresponds to Chapter 5]

Awards

1. 第 55 回自動制御連合講演会 優秀発表賞 (2012 年 12 月)
2. 平成 25 年度奈良先端科学技術大学院大学優秀学生 (2013 年 7 月)
3. IEEE Robotics and Automation Society Japan Chapter Young Award (2014 年 9 月)
4. RSJ/KROS Distinguished Interdisciplinary Research Award (2015 年 9 月)

Domestic Conference Proceedings

1. 田中大介, 松原崇充, 杉本謙二:
“入出力多様体学習による非線形システム同定”
第 55 回自動制御連合講演会 , pp. 21–25, 京都大学 吉田南キャンパス, 2012 年 11 月 17 日–18 日.
[Corresponds to Chapter 3]

2. 田中大介, 松原崇充, 杉本謙二:
“触覚・行動情報に基づく物体の多様体学習と能動的物体識別への応用”
第 26 回 自律分散システム・シンポジウム (DAS2014), 1B1-2, pp. 61–66 (CD-ROM), 東京大学 山上会館, 2014 年 1 月 23 日–24 日.
[Corresponds to Chapter 5]

3. 田中大介, 松原崇充, 杉本謙二:
   "触覚情報に基づく能動的物体認識のための探索行動設計 — 相互情報量を規範とする最適制御によるアプローチ —"
   第 32 回 日本ロボット学会学術講演会 (RSJ2014), 2I1-04 (DVD), 九州産業大学, 2014 年 9 月 4 日–6 日.
   [Corresponds to Chapter 5]

Acknowledgements

本論文を審査して頂きました奈良先端科学技術大学院大学 情報科学研究科 知能システム制御研究室 杉本謙二教授、ロボティクス研究室 小笠原司教授、知能システム制御研究室 松原崇充准教授、同研究室 南裕樹助教、岡山大学 平田健太郎教授に厚く御礼申し上げます。

杉本教授には博士前期の 1 年間を直接ご指導いただき、またその後も前期・後期課程を通して、ご指導をいただきました。ここに深く感謝いたします。

小笠原教授には、本研究の意義を主張する際に有益となるような様々な面からのご指摘をいただきました。その内容は本研究を進める上での指針となりました。改めて御礼申し上げます。

松原准教授には、前期課程の 2 年目から直接ご指導をいただきました。機械学習の基礎、論文の読み方から書き方に至るまで、様々な知識や作法をご指導・ご鞭撻賜りました。心より感謝申し上げます。

南助教には普段の雑談をはじめ、定例研究会などで鋭い質問や助言を多数いただき、本論文を執筆する際に参考とさせていただきました。改めて御礼申し上げます。

岡山大学 平田健太郎教授には前期課程在学時には特に密接に、日々の雑談や定例会を通して的確なアドバイスをいただきました。厚くお礼申し上げます。

また、福岡大学 野田賢教授（知能システム制御研究室 元准教授）には、著者が博士前期課程に在学中に定例研究会等でご助言をいただきました。深く感謝致します。電気通信大学 小木曽公尚准教授（知能システム制御研究室 元助教）には、他分野の視点から有益な助言を頂きました。深く感謝いたします。

研究・学生生活に関わる事務手続きを代行していただいた、秘書の林英子様、橋本洋子様に厚く御礼申し上げます。

知能システム制御研究室の諸先輩方や同期生の皆様、後輩の皆様にはときに励ましの言葉をいただきました。ありがとうございました。

最後に、学生生活を支援してくれた家族に感謝の意を表します。

Contents

List of Figures

List of Tables

Notation

Notation	Meaning
$\mathbb{R}$	set of real numbers
$\mathbf{I}_n$	$n \times n$-dimensional identity matrix*
$\mathbf{0}_n$	n-dimensional column vector filled with 0*
$\mathbf{O}_{n\times m}$	$n \times m$-dimensional zero matrix*
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	(multivariate) Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{N}(\mathbf{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	(multivariate) Gaussian distribution in $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{GP}(m(\cdot), k(\cdot,\cdot))$	Gaussian process with mean function $m(\cdot)$ and covariance (kernel) function $k(\cdot,\cdot)$
$\{\mathbf{x}_n\}_{n=1}^N$	set containing $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
$\mathbf{A}^\mathrm{T}$	transpose of the matrix $\mathbf{A}$
$\dot{x}$	time derivative
$\dot{\mathbf{x}}$	vector consisting of time derivative of each entry $[\dot{x}_1, \dots, \dot{x}_n]^\mathrm{T}$
$\mathbf{A}(:, k : l)$	submatrix of $\mathbf{A}$ consisting of columns $k, k+1, \dots, l$
$\mathbf{A}(i : j, k : l)$	submatrix of $\mathbf{A}$ consisting of the entries on rows $i, i+1, \dots, j$ and columns $k, k+1, \dots, l$
$\mathrm{Tr}(\cdot)$	trace
$\det(\cdot)$	determinant
$\mathrm{vec}(\cdot)$	vectorize operator
$\mathrm{diag}\{a, b, \dots\}$	diagonal matrix whose entry is $a, b, \dots$
$\mathrm{diag}\{\mathbf{A}, \mathbf{B}, \dots\}$	block diagonal matrix whose block entry is $\mathbf{A}, \mathbf{B}, \dots$
$\lambda_{\min}(\mathbf{A})$	the minimum eigenvalue of $\mathbf{A}$

(continued from previous page)

Notation	**Meaning**
$\lambda_{\max}(\mathbf{A})$	the maximum eigenvalue of $\mathbf{A}$
$\otimes$	Kronecker product
$\mathrm{sign}(\cdot)$	sign function
$\mathbf{A} \mathop{/}\limits_{\mathbf{B}} \mathbf{C}$	oblique projection of the row space $\mathbf{A}$ along the row space of $\mathbf{B}$ on the row space of $\mathbf{C}$

*If the dimension is not indicated, the size of the matrix or vector should be clear from the context.

Chapter 1

Introduction

This chapter introduces the overview of the dissertation. The research background is described in section 1.1. Next, the research motivation and the objective is presented in section 1.2. Then, the outline of this thesis is shown in section 1.3.

1.1. Background

Recently, various data from sensors are available in electronic form, and an intelligent system with these sensors is constructed to achieve many tasks. For

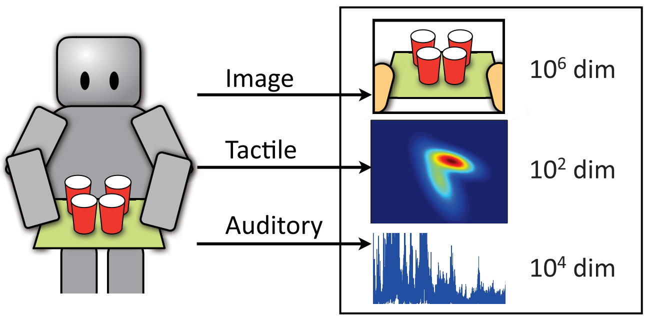


Figure 1.1. Robot as an example of a system. Since the amount of the data in one sample obtained from external sensor such as an image sensor (camera), a tactile sensor (pressure) and an auditory sensor (microphone) is large, the vectorized data will be high dimensional.

example, to enable automatic driving with intelligent mobility or to achieve efficient use of energy with a smart living environment, these systems (i.e., a car or a house) are employed with many sensors. Further, a robot is used for the assembly of parts, and it can be regarded as one of these systems. The robot is composed of external sensors such as camera, microphone, and pressure sensors in order to obtain the information around the robot (Fig. 1.1). In addition, the robot contains the internal sensors such as encoders, tachogenerators, and torque sensors in order to obtain the information that is related to the inside of the robot body. The sensor data of the robot is utilized for help in the nursing care and welfare units, and can be used to control the robot autonomously.

The obtained data is first used for modeling the generative models. For example, to achieve the assembly of parts with robots, the robot should be controlled accurately, and the robot should be able to recognize the required parts in order to pick up the part. The model-based control can be applied if the dynamical model of the robot is made available. Moreover, the classification of parts can be achieved with the generative model of the sensor model. However, the obtained data from the system is high dimensional. More concretely, since the amount of data in one sample is large, it will be high dimensional if whole data of each sample is vectorized. Modeling with such high-dimensional data is affected by the curse of dimensionality. To avoid the curse, the dimensionality reduction methods are proposed in the machine learning field, and the effectiveness is shown in many applications.

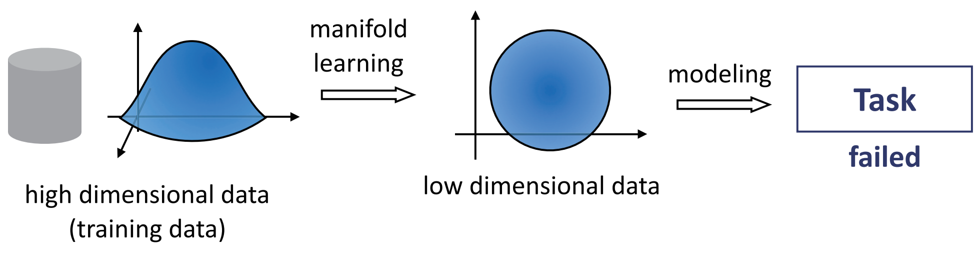
The dimensionality reduction methods reduce the dimensionality of the data without loss of the information under some assumptions. The principal component analysis (PCA) [3] is a well-known algorithm in multivariate statistics, and it is used for the visualization of the high-dimensional data. In PCA, it is assumed that the (observed) high-dimensional data is a linear projection of the low-dimensional data. The methods that can be regarded as similar methods to PCA are MultiDimensional Scaling (MDS) [4] and Probabilistic Principal Component Analysis (PPCA) [5].

However, if the assumption is not held (e.g. the nonlinear projection), applying a linear method such as PCA loses information. This deteriorates the task performance with the low-dimensional data. In contrast with the linear method, the nonlinear dimensionality reduction methods, sometimes called manifold learning methods, were proposed in 2000s. As typical methods, Isometric Feature Mapping (Isomap) [6], Locally Linear Embeddings (LLE) [7], and Laplacian Eigenmaps [8] are well known. One of the assumptions in Isomap is that the observation process is isometric, which is embedded in high dimensions. This assumption leads to failure in reduction of dimensionality without loss of information, if the manifold in the high-dimensional space is nonconvex. However, such a nonlinear dimensionality reduction method leads to success in reducing the dimensionality in many applications. Moreover, these manifold learning methods can be regarded as the kernel principal component analysis with a data-driven kernel. In the late of 2000s, the Gaussian Process Latent Variable Models (GPLVMs) [2] have been proposed as an extension of PPCA to nonlinear projection, and they successfully achieve the reduction of dimensionality even if the observation is noisy.

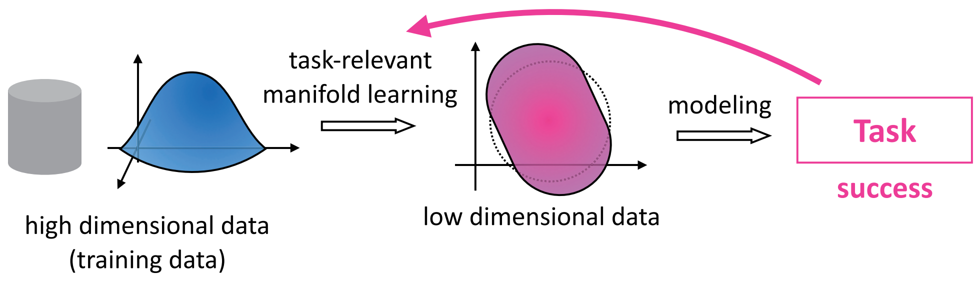
1.2. Research motivation and objective

Most of the dimensionality reduction methods are classified as unsupervised learning methods. The high-dimensional observation is only available when the low-dimensional data is computed. The criterion to find the low-dimensional data is based on some a prior knowledge (e.g. the observation was generated through linear generative model, or the space is convex). If the prior knowledge correctly fits to the data, the low-dimensional data can be obtained without information loss. However, the obtained low-dimensional data is not supposed to be used for system modeling for the task that is to be executed later, and so there is an open question, "are these criteria always the best for obtaining the low-dimensional data used for the task?". If the obtained low-dimensional data is not suitable for the task, then, the task fails (Fig. 1.2(a)).

In this dissertation, we propose task-relevant manifold learning methods. The task to be executed later is considered when the manifold is learned. The requirement is, for example, the model is linear or smooth. This requirement is converted into a new criterion or some additional constraints (Fig. 1.2(b)). Then, by solving the new optimization problem, the task-relevant manifold learning is achieved. Both the modeling and manifold learning are simultaneously achieved.



(a) Conventional: manifold learning without considering the task



task-relevant manifold learning
modeling
Task
success
high dimensional data (training data)
low dimensional data

(b) Proposed: with a task-relevant manifold learning

Figure 1.2. Illustration of the contribution of this dissertation

The proposition is validated through the following two problems.

First, we consider modeling using the high-dimensional data. The obtained model is useful for the analysis, or prediction. Particularly, we consider modeling of the linear dynamical systems represented using transfer function models. Thus, the task to be considered is to estimate the transfer function model using the high-dimensional data. In order to obtain an accurate model, we propose *input-output manifold learning with transfer function models* (IOMLTF) constructed by changing the criterion in the manifold learning. Through numerical simulations with synthetic data, the effectiveness of the proposed method will be validated.

Next, in order to consider the non-zero initial state of the system, we study another version of input-output manifold learning. Our solution is to use the state space model for the system representation. We call the proposed method *input-output manifold learning with state space models* (IOMLSS). That is, the state space model allows us to assume non-zero initial state. Additionally, more complicated noise models can be identified. The effectiveness of the proposed method will be validated with numerical simulations.

Lastly, a partial manifold learning method is considered in order to determine a suitable manifold to obtain the desired model structure. We consider a two-factor generative model whose one factor is known and the other is unknown. In order to obtain the unknown factor considering the effect of both factors to the observation, we propose a partial manifold learning method using the observation and the known factor. The method is based on GPLVMs and constructed by adding a constraint related to the desired model structure to the optimization problem. Further, this method is applied to obtain the suitable sensor model for object recognition using robots with tactile sensors. We call the proposed method *object manifold learning* in this particular application. The object recognition is achieved by sequential parameter estimation: first, a suitable parameter is allocated for each object, and the parameter is sequentially estimated by the sensor data obtained through an exploratory action of the robot. The requirements are 1) an intrinsic parameter should be allocated for each object since multiple observations are obtained by executing multiple exploratory actions to only one object, and 2) the model should be smooth to make the problem easier. Thus, the first consideration is to estimate the suitable parameter for each object from the observation considering the effect of the action and the smoothness of the parameter in the model. The proposed manifold learning method will be applied to this problem to extract the object manifold which lies on the tactile sensor space. In addition, particularly considering the object recognition problem, the quality of the obtained data affects to the quickness of the recognition. In other words, the informativeness of the obtained data is important. Simultaneously, the compliance of the exploratory action should be considered to avoid breaking the object and the robot. Hence, we additionally need to consider obtaining informative and compliant exploratory action. To achieve this, we propose *information maximization control*. The information maximization control then generates the informative and compliant action based on the optimal control framework. The effectiveness of the proposed methods is validated through the numerical simulation with the synthetic data and obtained data from the actual robot.

1.3. Dissertation layout

This dissertation is composed of chapters as described below. The flow of the chapters is shown in Fig. 1.3.

Chapter 2 provides an overview of the manifold learning methods, which form the base of the proposed method as shown in Fig. 1.3. The dimensionality reduction schemes with linear generative models such as PCA and MDS are described. Then, Isomap, which is a nonlinear dimensionality reduction scheme, is explained as an extension of MDS. Finally, the dimensionality reduction methods with probabilistic generative models such as PPCA and GPLVMs are described.

Chapter 3 describes modeling from high dimensional data, namely input-output learning with transfer function model (IOMLTF). First, the manifold learning method is reformulated as a quadratic programming problem with quadratic constraint condition. Then, the changed criterion for the accurate model is described. The numerical simulation with synthetic data and image data (Fig. 1.4) used to validate the criterion is shown.

Chapter 4 describes the extension of the IOMLTF to capture the transient response, namely input-output learning with state space model (IOMLSS). The numerical simulation with synthetic data for validation is shown.

Chapter 5 describes a partial manifold learning method for two-factor generative models. In this chapter, we particularly consider the object recognition problem using a robot hand with tactile sensor shown in Fig. 1.5 as the application of the method. In this application, we call the proposed method object manifold learning. The proposed method is applied to obtain the probabilistic sensor model and suitable parameters of each object. In addition, to consider the compliance of the exploratory action, information maximization control is proposed. The proposed methods are validated through the numerical simulations with synthetic data and the obtained data from the actual robot hand.

Chapter 6 concludes this dissertation with discussion.

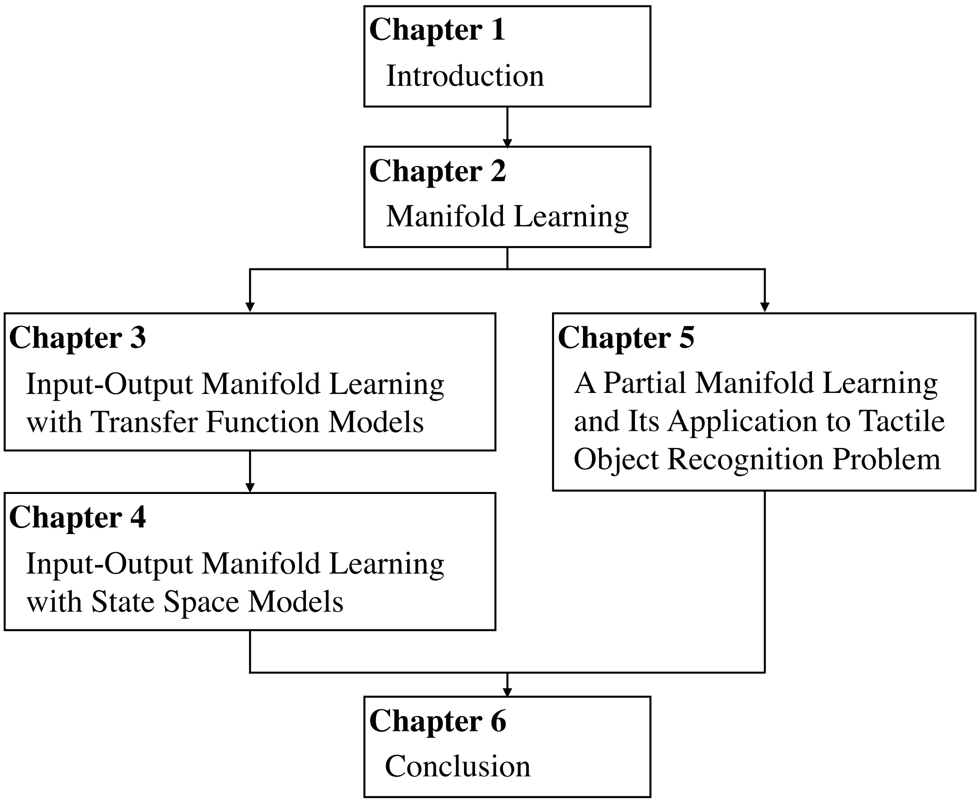

Chapter 1
Introduction
Chapter 2
Manifold Learning
Chapter 3
Input-Output Manifold Learning with Transfer Function Models
Chapter 5
A Partial Manifold Learning and Its Application to Tactile Object Recognition Problem
Chapter 4
Input-Output Manifold Learning with State Space Models
Chapter 6
Conclusion

Figure 1.3. Flow of the chapters of this dissertation.

Figure 1.4. Image used for validation in Chapter 3.

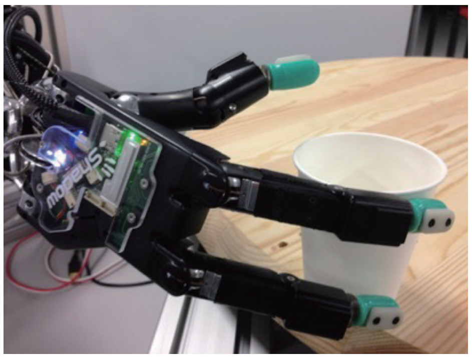


Figure 1.5. Anthropomorphic robot hand used in Chapter 5.

Chapter 2

Manifold Learning

In this chapter, we describe the dimensionality reduction methods based on generative models. We consider the following function

$$\mathbf{y} = \mathbf{f}(\mathbf{x}), \tag{2.1}$$

where $\mathbf{y} \in \mathbb{R}^{d_y}$ is the (known) d_y-dimensional observation, and $\mathbf{x} \in \mathbb{R}^{d_x}$ is the (unknown) d_x-dimensional latent variable. This model assumption indicates that the observation $\mathbf{y}$ is obtained as a function value of $\mathbf{x}$. Thus, such a model is called a *generative model*. Assuming $d_y > d_x$, the operation to find suitable $\mathbf{x}$ corresponding $\mathbf{y}$ is regarded as the dimensionality reduction of $\mathbf{y}$. Note that the function $\mathbf{f}$ is also unknown. Hence, it is important to determine a suitable function $\mathbf{f}$. We assume $N > d_y$ data set $\{\mathbf{y}_i\}_{i=0}^{N}$ is available, and its mean $\bar{\mathbf{y}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i$ is assumed to be zero. In this chapter, we consider the datasets shown in Fig. 2.1.

2.1. Linear dimensionality reduction

Let us assume a linear generative model:

$$\mathbf{y} = \mathbf{W}\mathbf{x}, \tag{2.2}$$

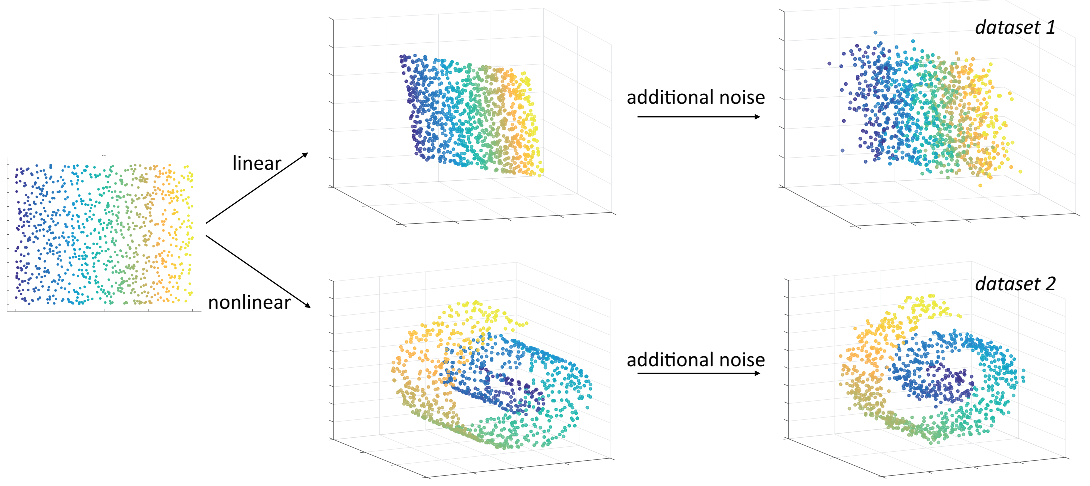


Figure 2.1. The two datasets used in this chapter. (Left): true low-dimensional data $\mathbf{x}$. (Upper Middle) and (Upper Right): observation using linear generative model and the observation with additional noise, respectively (dataset 1, noisy 3 dimensional data). (Lower Middle) and (Lower Right): observation using non-linear generative model and the observation with additional noise, respectively (dataset 2, noisy swissroll dataset).

where $\mathbf{W}$ is a $d_y \times d_x$ orthogonal matrix: $\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{I}_{d_x}$. Then, let us consider the average of the squared reconstruction error E:

$$E = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2 , \tag{2.3}$$

where $\hat{\mathbf{y}}_i = \mathbf{W}\mathbf{x}_i$. The model with the orthogonal matrix $\mathbf{W}$ and the corresponding latent variables $\mathbf{x}_i$ that minimize the error E defined by Eq. (2.3) is called *(classical) principal component analysis* (PCA).

The optimal solution of $\mathbf{W}$ is obtained by setting $\mathbf{W} = \mathbf{V}_{d_x}$, where $\mathbf{V}_{d_x}$ is composed of with the eigenvectors of the covariance matrix $\mathbf{\Sigma}_{yy} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i\mathbf{y}_i^{\mathrm{T}}$ associated with d_x eigenvalues in decreasing order. Let us describe the derivation. Firstly, we define the matrix $\mathbf{Y} \in \mathbb{R}^{N\times d_y}$ as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \end{bmatrix}^{\mathrm{T}} .$$

Then, the matrix can be decomposed by singular value decomposition as follows:

$$\mathbf{Y} = \mathbf{U}_Y \mathbf{S}_Y \mathbf{V}_Y^{\mathrm{T}}$$

where $\mathbf{U}_Y$ is the $N \times N$ matrix with orthogonal columns ($\mathbf{U}_Y^{\mathrm{T}}\mathbf{U}_Y = \mathbf{I}_N$), $\mathbf{S}_Y$ is the $N \times d_y$ matrix, $\mathbf{V}_Y$ is the $d_y \times d_y$ orthogonal matrix ($\mathbf{V}_Y^{\mathrm{T}}\mathbf{V}_Y = \mathbf{V}_Y\mathbf{V}_Y^{\mathrm{T}} = \mathbf{I}_{d_y}$), and $\mathbf{S}_Y$ is the $N \times d_y$ matrix whose (i,i)-entry contains the singular values $s_i \geq 0$ for $i = 1, \ldots, d_y$. It can be shown that $\mathbf{U}_Y$ is associated with the eigenvectors of $\mathbf{Y}\mathbf{Y}^{\mathrm{T}}$. Using this decomposition, we can obtain a rank d_x approximation as follows:

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \hat{\mathbf{U}}_Y \hat{\mathbf{S}}_Y \hat{\mathbf{V}}_Y^{\mathrm{T}}$$

where

$$\hat{\mathbf{U}}_Y = \mathbf{U}_Y(\,:\,, 1\!:\!d_x) \tag{2.4}$$

$$\hat{\mathbf{S}}_Y = \mathbf{S}_Y(1\!:\!d_x, 1\!:\!d_x) \tag{2.5}$$

$$\hat{\mathbf{V}}_Y = \mathbf{V}_Y(\,:\,, 1\!:\!d_x).$$

The generative model (2.2) indicates that the relationship between all observation and corresponding latent variables is represented by $\mathbf{Y} = \mathbf{X}\mathbf{W}^{\mathrm{T}}$. Then, using the approximated matrix $\hat{\mathbf{Y}}$, the latent variables $\mathbf{X}$ and the matrix $\mathbf{W}$ are obtained as follows

$$\hat{\mathbf{X}} = \hat{\mathbf{U}}_Y \hat{\mathbf{S}}_Y, \tag{2.6}$$

$$\hat{\mathbf{W}} = \hat{\mathbf{V}}_Y. \tag{2.7}$$

It can be shown that the matrix $\hat{\mathbf{V}}$ is equal to $\mathbf{V}_{d_x}$.

Let us introduce another method of using the linear generative model. We consider a matrix $\mathbf{D}_Y$ whose (i,j)-entry consists of the Euculid distance between the observations,

$$\begin{aligned} D_{Y,ij} = \|\mathbf{y}_i - \mathbf{y}_j\|^2 &= \|\mathbf{y}_i\|^2 + \|\mathbf{y}_j\|^2 - 2\mathbf{y}_i^{\mathrm{T}}\mathbf{y}_j \\ &= K_{Y,ii} + K_{Y,jj} - 2K_{Y,ij}, \end{aligned} \tag{2.8}$$

where $K_{Y,ij}$ is the (i,j) entry of $\mathbf{K}_Y$. Since the mean of the observations $\bar{\mathbf{y}}$ is $\mathbf{0}$, the following relationship is satisfied:

$$\sum_{i=1}^{N} \mathbf{y}_i^{\mathrm{T}}\mathbf{y}_j = 0,$$

for $j = 1, \dots, N$. Using this equation and Eq. (2.8), the matrix $\mathbf{K}_Y$ can be obtained using $\mathbf{D}$ as follows:

$$\mathbf{K}_Y = -\frac{1}{2}\mathbf{H}\mathbf{D}_Y\mathbf{H}, \tag{2.9}$$

where $\mathbf{H}$ is the centering matrix $\mathbf{H} = \mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^{\mathrm{T}}$. We can rewrite the expression as $\mathbf{K}_Y = \mathbf{Y}\mathbf{Y}^{\mathrm{T}} = \mathbf{X}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{X}^{\mathrm{T}} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$. Then, the low-dimensional latent variables are obtained through the eigen decomposition of $\mathbf{K}_Y = \mathbf{U}_{K_Y}\mathbf{S}_{K_Y}\mathbf{U}_{K_Y}^{\mathrm{T}}$,

$$\hat{\mathbf{X}} = \hat{\mathbf{U}}_{K_Y}\hat{\mathbf{S}}_{K_Y}^{1/2}, \tag{2.10}$$

where $\hat{\mathbf{U}}_{K_Y}$ and $\hat{\mathbf{S}}_{K_Y}$ are composed of d_x eigenvectors and the corresponding d_x eigenvalues of $\mathbf{K}_Y$ in decreasing order respectively. This is same as Eq. (2.6). The result indicates that the low-dimensional latent variables can be obtained from the distance among observations, and this method is called MultiDimensional Scaling (MDS). Note that the MDS does not output the matrix $\mathbf{W}$. Also note that the latent variables $\mathbf{X}$ can be obtained from the eigen decomposition of $\mathbf{K}_Y$ without scaling.

The results for the datasets are shown in Fig. 2.2. As shown in the figure,

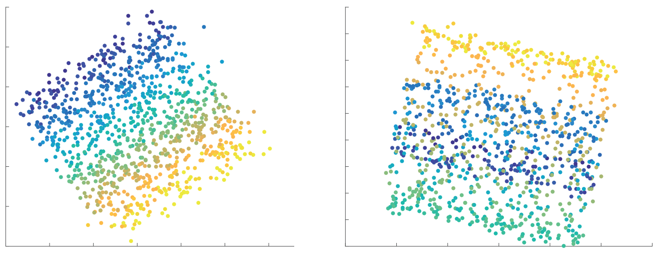


Figure 2.2. Dimensionality reduction results with PCA. (Left) Results for dataset 1. (Right) Results for dataset 2. PCA determines the low-dimensional structure well for dataset 1. However, the roll structure is squashed for dataset 2. Hence, the nonlinear structure cannot be found by PCA.

PCA determines the low-dimensional structure well for dataset 1. However, the roll structure is squashed for dataset 2. Hence, the nonlinear structure is difficult to determine using PCA.

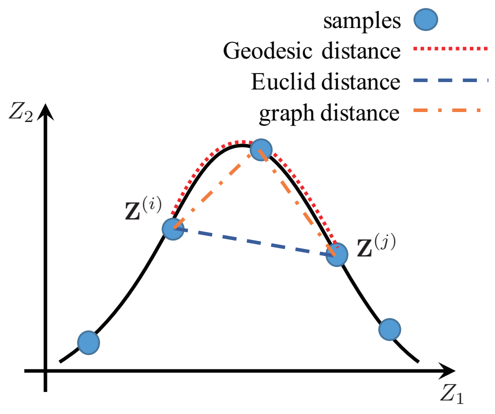


Figure 2.3. Comparison of the distances. The black line shows a one-dimensional manifold in two-dimensional space. The objective is to determine the appropriate one-dimensional representation from the obtained samples. MDS can be used to determine the low-dimensional representation such that the Euclid distance among high-dimensional samples holds as much as possible. Isomap focuses on the geodesic distance instead, as the original structure is nonlinear. The geodesic distance is approximated using the graph distance.

2.2. Nonlinear dimensionality reduction

Additionally, let us consider the feature vector of $\mathbf{y}$ as follows,

$$\mathbf{z} = \boldsymbol{\phi}(\mathbf{y}).$$

Hence, the relationship between $\mathbf{x}$ and $\mathbf{z}$ is nonlinear. When we construct $\mathbf{K}_Z$ similar to Eq. (2.9), we need to consider the distances between $\boldsymbol{\phi}(\mathbf{y}_i)$ and $\boldsymbol{\phi}(\mathbf{y}_j)$. By defining $\boldsymbol{\phi}$ so that the distance among $\{\boldsymbol{\phi}(\mathbf{y}_i)\}_{i=1}^N$ corresponds to a graph distance among $\{\mathbf{y}_i\}_{i=1}^N$, which is the approximation of the geodesic distance (see Fig. 2.3, and see Appendix A for the computation method of the graph distance), we can determine $\{\mathbf{x}_i\}_{i=1}^N$, and the method is known as isometric feature mapping (Isomap).

We can find a nonlinear structure using Isomap even it is difficult to find using PCA and MDS. Isomap can be regarded as an extension of MDS. It is assumed that the squared graph distance between the i-th sample and j-th sample $d_z^2(i,j)$

for $i, j = 1, \ldots, N$ is stored in the $N \times N$ matrix $\mathbf{D}_z$ as follows:

$$(\mathbf{D}_z)_{ij} = d_z^2(i, j).$$

Then, with an assumption that the samples are centered (zero-mean), the matrix $\mathbf{K}_z$ can be constructed using $\mathbf{D}_z$ in a similar way to Eq. (2.10) as

$$\mathbf{K}_z = -\frac{1}{2}\mathbf{H}\mathbf{D}_z\mathbf{H}$$

where $\mathbf{H}$ is the centering matrix $\mathbf{H} = \mathbf{I}_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^{\mathrm{T}}$. The only difference is the distance used for $\mathbf{D}$. MDS is regarded as a special case where $\boldsymbol{\phi}(\mathbf{y}) = \mathbf{y}$. With both distances, the low-dimensional representation can be obtained by solving the eigenvalue decomposition problem. The nonlinear mapping can be successfully found with Isomap as shown in Fig. 2.4. However, some data points are mixed because of the observation noise.

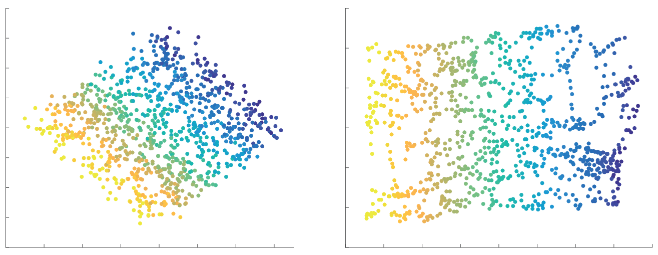


Figure 2.4. Dimensionality reduction results with Isomap. (Left) Results for dataset 1. (Right) Results for dataset 2. For both dataset 1 and dataset 2, Isomap can be used to determine the suitable low-dimensional representation.

2.3. Dimensionality reduction with probabilistic model

To take the observation noise into account, let us consider the following generative model:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \boldsymbol{\epsilon} \tag{2.11}$$

where $\boldsymbol{\epsilon}$ is the Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Let us represent this model using a probability distribution as follows,

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{W}\mathbf{x}, \sigma^2\mathbf{I}) \tag{2.12}$$

The prior of the latent variable $p(\mathbf{x})$ is set to a Gaussian $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, the marginal distribution $p(\mathbf{y})$ can be obtained as

$$\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\mathbf{x} \\ &= \mathcal{N}(\mathbf{0}, \mathbf{C}) \end{aligned}$$

where the covariance matrix $\mathbf{C}$ is defined as

$$\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}.$$

Let us determine $\mathbf{W}$ and σ^2 using the maximum likelihood method. The corresponding log-likelihood is obtained as

$$\begin{aligned} \ln p(\mathbf{Y}|\mathbf{W}, \sigma^2) &= \sum_{i=1}^{N} \ln p(\mathbf{y}_i|\mathbf{W}, \sigma^2) \\ &= -\frac{Nd_y}{2}\ln(2\pi) - \frac{N}{2}\ln\det\mathbf{C} - \frac{1}{2}\sum_{i=1}^{N}\mathbf{x}_i^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{x}_i \\ &= -\frac{N}{2}\{D\ln(2\pi) + \ln\det\mathbf{C} + \mathrm{Tr}(\mathbf{C}^{-1}\mathbf{K}_Y)\}. \end{aligned}$$

The optimal $\mathbf{W}$ and σ^2 are obtained as follows:

$$\begin{aligned} \mathbf{W}_{\mathrm{ML}} &= \hat{\mathbf{U}}_Y(\hat{\mathbf{S}}_Y - \sigma^2\mathbf{I})^{1/2} \\ \sigma^2_{\mathrm{ML}} &= \frac{1}{d_y - d_x}\sum_{i=d_x+1}^{d_y} s_i \end{aligned}$$

where $\hat{\mathbf{U}}_Y$ and $\hat{\mathbf{S}}_Y$ are defined by Eqs. (2.4) and (2.5) respectively. This is known as probabilistic PCA (PPCA). By setting $\sigma^2 \to 0$, the maximum likelihood solution of the PPCA is reduced to the solution of PCA.

Let us consider a dual problem: obtain $\mathbf{X}$ instead of $\mathbf{W}$. First, we define the prior over $\mathbf{W}$ as follows

$$p(\mathbf{W}) = \prod_{j=1}^{d_y} \mathcal{N}(\mathbf{w}_j|\mathbf{0}, \mathbf{I}).$$

Then, the log-likelihood of the marginal distribution $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{W})p(\mathbf{W})\mathrm{d}\mathbf{W}$ becomes

$$\ln p(\mathbf{Y}|\mathbf{X}, \sigma^2) = -\frac{N}{2}\{D\ln(2\pi) + \ln\det\mathbf{K}_X + \mathrm{Tr}(\mathbf{K}_X^{-1}\mathbf{Y}\mathbf{Y}^{\mathrm{T}})\},$$

where

$$\mathbf{K}_X = \mathbf{X}\mathbf{X}^{\mathrm{T}} + \sigma^2\mathbf{I}. \tag{2.13}$$

Now, let us consider other representation of $\mathbf{K}_X$ as follows,

$$\mathbf{K}_X = \mathbf{K} + \sigma^2\mathbf{I}.$$

If $\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$, it is same as Eq. (2.13). In this case, the (i, j)-entry of $\mathbf{K}$ is $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$. Replacing $\mathbf{x}_i$ with $\boldsymbol{\psi}(\mathbf{x}_i)$ and defining $k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\psi}^{\mathrm{T}}(\mathbf{x}_i)\boldsymbol{\psi}(\mathbf{x}_j)$, the (i, j)-entry of $\mathbf{K}$ is replaced with $k(\mathbf{x}_i, \mathbf{x}_j)$. The function $k(\mathbf{x}, \mathbf{x}')$ is called a (Mercer) kernel function. The vector $\boldsymbol{\psi}$ is implicitly determined by the function k. In the case of Eq. (2.13), as $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$, the kernel is called a linear kernel. One of other selections is a squared exponential kernel function expressed as:

$$k(\mathbf{x}, \mathbf{x}') = \alpha^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^{\mathrm{T}}\mathbf{M}^{-1}(\mathbf{x} - \mathbf{x}')\right)$$

where $\mathbf{M}$ is a diagonal matrix with positive elements. The likelihood function with this kernel function is determined with $\gamma = \{\alpha^2, \mathbf{M}, \sigma^2\}$. Thus, γ is called a hyperparameter. The maximum likelihood solution is no longer obtained by eigen decomposition. Instead, we can obtain the solution via numerical optimization methods, e.g. $\max_{\mathbf{X},\gamma} \ln p(\mathbf{Y}|\mathbf{X}, \gamma)$. The dimensionality reduction result with GPLVMs is shown in Fig. 2.5. The initial low-dimensional data are obtained through Isomap, and the hyperparameters and low-dimensional data are simultaneously optimized using a quasi-newton algorithm. The data points have enough distance against with the observation noise, as shown in the figure.

2.4. Summary

We described the manifold learning methods. The properties of each method are summarized in Table 2.1. A comparison of the methods can be found in [9, 10].

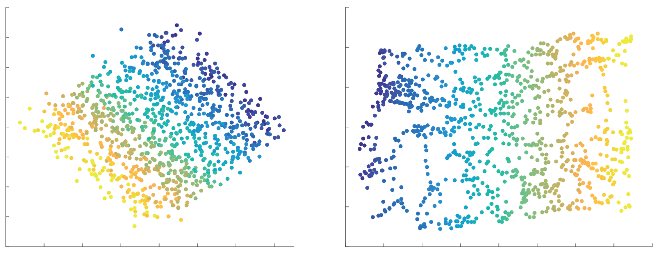


Figure 2.5. Dimensionality reduction results with GPLVMs. (Left) Results for dataset 1. (Right) Results for dataset 2. Each point of low-dimensional data is separated, as shown in the left figure of Fig. 2.1.

Table 2.1. Comparison of the methods described in this chapter.

Method	Generative model	Criterion
PCA	Deterministic, linear	Reconstruction error
MDS	Deterministic, linear	Euclid distance error
Isomap	Deterministic, nonlinear	Graph distance error
PPCA	Probabilistic, linear	Marginal log-likelihood
GPLVM	Probabilistic, nonlinear	Marginal log-likelihood

Chapter 3

Linear System Identification from High-dimensional Data

3.1. Introduction

The sensor data such as image, sound and tactile can be easily obtained given that good and cheap sensors are available nowadays. The obtained data are useful for many objectives such as recognition, prediction, and control. By analyzing multiple data sets simultaneously, we can find the latent relationship among the data sets. These data are often obtained as high-dimensional vectors. For example, a 128×128 pixel RGB image is regarded as a 49,152 ($= 128 \times 128 \times 3$) dimensional vector. This study focuses on the system identification problem that arises when applying such high-dimensional data for both input and output.

In general, the raw data obtained from the sensors may not be used directly owing to its high dimensionality, which induces high computational cost. Instead, a suitable feature is extracted from data for the purpose of dimensionality reduction of data. As a hand-tuned feature, for the tracking objective, the centroid position of a specific area in the obtained image is calculated. As a data-driven feature, SIFT or HOG features are used for image-based recognition tasks (e.g. [11, 12]). Manifold learning methods such as Isomap [6], LLE [7], or GPLVM [2] can be used as data-driven feature extractors for more general objectives. The effectiveness of these schemes for various applications such as the video image [13], and motion capture data [14] has been shown.

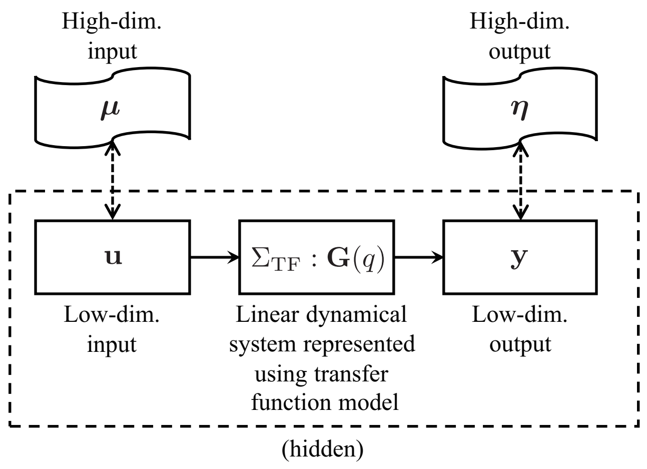


Figure 3.1. Problem setting of IOMLTF. The obtained data are high-dimensional input $\boldsymbol{\mu}$ and output $\boldsymbol{\eta}$. It is assumed that the features regarded as low-dimensional input $\mathbf{u}$ and output $\mathbf{y}$ have the relationship represented as a linear dynamical system Σ_{TF}, represented using a transfer function model.

Let us assume that the relationship between the features extracted from two high-dimensional data is represented as a linear time-invariant dynamical system. The approach to estimate the system based on from input and output data will first require the application of the manifold learning method individually for two data, and then the system is estimated by using a system identification method. However, given that the manifold learning methods are unsupervised, the features extracted from the data cannot be exactly the same as the true features: therefore, the estimated latent relationship between them may not be represented by a LTI system anymore. This means that a simple linear system identification problem is undesirably converted to complex nonlinear system identification problem.

To resolve this issue, we propose a paradigm that simultaneously considers the feature extractions by manifold learning and linear system identification on the extracted feature space. To realize this concept, we propose the input-output manifold learning with transfer function model (IOMLTF). Figure 3.1 illustrates this problem setting. First, the IOMLTF assumes the linear system is represented using a transfer function model. The features (referred to as low-dimensional input and output) of high-dimensional data (referred to as high-dimensional input and output) are extracted by Isomap [6] with a regularization term that considers the fitness for the linear system. Then, linear system identification is applied to update the transfer function model. Repeated application of these optimization steps converges to a locally optimal solution.

The remainder of this chapter is organized as follows. Firstly, the related work is listed in Section 3.2. Our proposition, IOMLTF is described in Section 3.3. The numerical simulation for validating IOMLTF is shown in section 3.4, and the summary and the scope for future work are discussed in Section 3.5.

3.2. Related works

With regard to related works, Weight Determination by Manifold Regularization [15] has been proposed. This method regards the system identification problem as a regression problem between the input and output, and it finds the low dimensional input considering the regression performance. However, finding the low dimensional output is not discussed in the method. Our problem setting is difficult to be applied. In addition, Gaussian process dynamical models [16] or variational Gaussian process dynamical systems [17], which are the methods to find the latent dynamics in the observations, seem to be applicable to our problem settings. Nevertheless, the exogenous input is not considered in the methods. In addition, the linear dynamical system may not be considered directly.

3.3. Proposed method: input-output manifold learning with transfer function model

3.3.1 Problem settings

Now, let us consider an unknown linear time-invariant discrete-time dynamical system Σ with the following input and output

$$\begin{aligned}\mathbf{u}(t) &:= [u_1(t), \dots, u_{d_u}(t)]^{\mathrm{T}} \in \mathbb{R}^{d_u},\\ \mathbf{y}(t) &:= [y_1(t), \dots, y_{d_y}(t)]^{\mathrm{T}} \in \mathbb{R}^{d_y},\end{aligned}$$

which are referred to as the low-dimensional input and output of the system, respectively. The objective is to find this relationship from the high-dimensional input $\boldsymbol{\mu}(t) \in \mathbb{R}^{d_\mu}(d_\mu > d_u)$ and output $\boldsymbol{\eta}(t) \in \mathbb{R}^{d_\eta}(d_\eta > d_y)$. Here, we assume that the dataset of the high-dimensional input and output from time $t = 0$ to $t = T-1$, $\mathcal{D} = \{\boldsymbol{\mu}(t), \boldsymbol{\eta}(t)\}_{t=0}^{T-1}$, is available, and the system order n and the dimensionality of the low-dimensional input d_u and output d_y are known. Figure 3.1 illustrates this problem setting.

3.3.2 Iterative optimization scheme

Let us define the matrices comprising the data from $t = 0$ to $t = T - 1$ as follows:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^{\mathrm{T}}(0) \\ \vdots \\ \mathbf{u}^{\mathrm{T}}(T-1) \end{bmatrix} =: \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{d_u} \end{bmatrix},$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{\mathrm{T}}(0) \\ \vdots \\ \mathbf{y}^{\mathrm{T}}(T-1) \end{bmatrix} =: \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_{d_y} \end{bmatrix}.$$

The system Σ could be identified from $\mathbf{U}$ and $\mathbf{Y}$. However, all of these variables are unknown, and simultaneous estimation of all unknown variables can pose a difficulty. Therefore, we propose the solution methods of two optimization problems called *input-output manifold learning* as an efficient algorithm.

Obtaining the low-dimensional representation

Assuming that the system Σ is given, we first solve the manifold learning problem, which is reconstructed by adding the penalty term so that the low-dimensional data follow the dynamics:

$$\min_{\mathbf{U},\mathbf{Y}} f(\mathbf{U}, \mathbf{Y}, \Sigma, \mathbf{K}_\mu, \mathbf{K}_\eta, \gamma) \qquad \text{s.t. } \mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_{d_u},\ \mathbf{Y}^{\mathrm{T}}\mathbf{Y} = \mathbf{I}_{d_y} \tag{3.1}$$

where

$$f(\mathbf{U}, \mathbf{Y}, \Sigma_{\mathrm{TF}}, \mathbf{K}_\mu, \mathbf{K}_\eta, \gamma) := \gamma f_1(\mathbf{U}, \mathbf{Y}, \mathbf{K}_\mu, \mathbf{K}_\eta) + (1-\gamma) f_2\big(\mathbf{U}, \mathbf{Y}, \Sigma_{\mathrm{TF}}\big). \tag{3.2}$$

The function $f_1(\mathbf{U}, \mathbf{Y}, \mathbf{K}_\mu, \mathbf{K}_\eta)$ with the constraints in equation (3.1) is a function to find the eigenvectors of $\mathbf{K}_\mu$ and $\mathbf{K}_\eta$ which corresponds to the maximum eigenvalue,

$$f_1(\mathbf{U}, \mathbf{Y}, \mathbf{K}_\mu, \mathbf{K}_\eta) := \begin{bmatrix} \mathrm{vec}\big(\mathbf{U}\big) \\ \mathrm{vec}\big(\mathbf{Y}\big) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{I}_{d_u} \otimes \mathbf{K}_\mu^{-1} & \\ & \mathbf{I}_{d_y} \otimes \mathbf{K}_\eta^{-1} \end{bmatrix} \begin{bmatrix} \mathrm{vec}\big(\mathbf{U}\big) \\ \mathrm{vec}\big(\mathbf{Y}\big) \end{bmatrix}$$

which is identical to the original manifold learning methods. Here, the matrices $\mathbf{K}_\mu, \mathbf{K}_\eta$ are defined by the gram matrix of Isomap for each, and these are assumed to be positive definite. If the matrix is not positive definite, the following operation can be applied: $\mathbf{K}_* \leftarrow \mathbf{K}_* - \lambda_{\min}(\mathbf{K}_*)\mathbf{I}$.

In addition, the function $f_2\big(\mathbf{U}, \mathbf{Y}, \Sigma\big)$ represents the fitting error for the system. We consider a transfer function model as the system representation of Σ_{TF},

$$\Sigma_{\mathrm{TF}} : \mathbf{y}(t) = \mathbf{G}(q)\mathbf{u}(t).$$

Here, q is the shift operator $q^{-1}\mathbf{y}(t) = \mathbf{y}(t-1)$, $\mathbf{G}(q)$ is the transfer function matrix, and (i, j) entry of the matrix is the transfer function from the j-th input to the i-th output:

$$G_{ij}(q) = \frac{b_0^{(i,j)} + b_1^{(i,j)} q^{-1} + \cdots + b_n^{(i,j)} q^{-n}}{1 + a_1^{(i,j)} q^{-1} + \cdots + a_n^{(i,j)} q^{-n}},$$

for $i = 1, 2, \ldots, d_y$ and $j = 1, 2, \ldots, d_u$. The squared fitting error to this system is represented as follows:

$$\begin{aligned} f_2\big(\mathbf{U}, \mathbf{Y}, \Sigma_{\mathrm{TF}}\big) &= f_2^t\big(\mathbf{U}, \mathbf{Y}, \mathbf{G}(q)\big) && (3.3) \\ &:= \sum_{i=1}^{d_y} \sum_{j=1}^{d_u} \left\| \mathbf{W}_y^{(i,j)} \mathbf{y}_i - \mathbf{W}_u^{(i,j)} \mathbf{u}_j \right\|^2 . && (3.4) \end{aligned}$$

where the matrices $\mathbf{W}_y^{(i,j)}$ and $\mathbf{W}_u^{(i,j)}$ are asymmetric Toeplitz matrices represented as follows,

$$\mathbf{W}_y^{(i,j)} = \big[a_{m-n}^{(i,j)}\big]_{m,n=0}^{T-1}, \ \mathbf{W}_u^{(i,j)} = \big[b_{m-n}^{(i,j)}\big]_{m,n=0}^{T-1},$$

with $a_0^{(i,j)} = 1$, $a_k^{(i,j)} = b_k^{(i,j)} = 0$ for $k < 0$ or $k > n$.

The weighting factor $\gamma, (0 < \gamma \leq 1)$ is a free parameter, which can be manually modified. The function f can be represented in the quadratic form as follows:

$$f(\mathbf{U}, \mathbf{Y}, \Sigma, \mathbf{K}_\mu, \mathbf{K}_\eta, \gamma) = \begin{bmatrix} \mathrm{vec}(\mathbf{U}) \\ \mathrm{vec}(\mathbf{Y}) \end{bmatrix}^{\mathrm{T}} (\gamma \mathcal{K} + (1-\gamma)\mathcal{W}) \begin{bmatrix} \mathrm{vec}(\mathbf{U}) \\ \mathrm{vec}(\mathbf{Y}) \end{bmatrix}$$

where

$$\mathcal{K} := \begin{bmatrix} \mathbf{I}_{d_u} \otimes \mathbf{K}_\mu^{-1} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_{d_y} \otimes \mathbf{K}_\eta^{-1} \end{bmatrix},$$

$$\mathcal{W} := \begin{bmatrix} \mathcal{W}_{uu} & -\mathcal{W}_{uy} \\ -\mathcal{W}_{uy}^{\mathrm{T}} & \mathcal{W}_{yy} \end{bmatrix},$$

$$\mathcal{W}_{uu} = \begin{bmatrix} \mathbf{W}_u^{(1,1)^{\mathrm{T}}} \mathbf{W}_u^{(1,1)} & \cdots & \mathbf{W}_u^{(1,1)^{\mathrm{T}}} \mathbf{W}_u^{(d_u,d_u)} \\ \vdots & \ddots & \vdots \\ \mathbf{W}_u^{(d_u,1)^{\mathrm{T}}} \mathbf{W}_u^{(d_u,1)} & \cdots & \mathbf{W}_u^{(d_u,d_u)^{\mathrm{T}}} \mathbf{W}_u^{(d_u,d_u)} \end{bmatrix},$$

$$\mathcal{W}_{uy} = \begin{bmatrix} \mathbf{W}_u^{(1,1)^{\mathrm{T}}} \mathbf{W}_y^{(1,1)} & \cdots & \mathbf{W}_u^{(1,d_y)^{\mathrm{T}}} \mathbf{W}_y^{(1,d_y)} \\ \vdots & \ddots & \vdots \\ \mathbf{W}_u^{(d_u,1)^{\mathrm{T}}} \mathbf{W}_y^{(d_u,1)} & \cdots & \mathbf{W}_u^{(d_u,d_y)^{\mathrm{T}}} \mathbf{W}_y^{(d_u,d_y)} \end{bmatrix},$$

$$\mathcal{W}_{yy} = \begin{bmatrix} \mathbf{W}_y^{(1,1)^{\mathrm{T}}} \mathbf{W}_y^{(1,1)} & \cdots & \mathbf{W}_y^{(1,d_y)^{\mathrm{T}}} \mathbf{W}_y^{(1,d_y)} \\ \vdots & \ddots & \vdots \\ \mathbf{W}_y^{(d_y,1)^{\mathrm{T}}} \mathbf{W}_y^{(d_y,1)} & \cdots & \mathbf{W}_y^{(d_y,d_y)^{\mathrm{T}}} \mathbf{W}_y^{(d_y,d_y)} \end{bmatrix}.$$

If we assume a SISO system ($d_u = d_y = 1$), this optimization problem can be reformulated as a Multi-Eigenvalue Problem (MEP) [18] (see Appendix B). First, the optimization problem (3.1) is represented in the following quadratic form,

$$\min_{\mathbf{u}_1, \mathbf{y}_1} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \gamma \mathbf{K}_\mu^{-1} + (1-\gamma)\mathcal{W}_{uu} & -(1-\gamma)\mathcal{W}_{uy} \\ -(1-\gamma)\mathcal{W}_{uy}^{\mathrm{T}} & \gamma \mathbf{K}_\eta^{-1} + (1-\gamma)\mathcal{W}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_1 \end{bmatrix} \tag{3.5}$$
$$\text{s.t. } \mathbf{u}_1^{\mathrm{T}} \mathbf{u}_1 = 1, \mathbf{y}_1^{\mathrm{T}} \mathbf{y}_1 = 1$$

We apply the method of Lagrange multiplier. Let us introduce the auxiliary function,

$$\begin{aligned}
&\mathcal{L}(\mathbf{u}_1, \mathbf{y}_1, \lambda_u, \lambda_y) \\
&\quad = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \gamma \mathbf{K}_\mu^{-1} + (1-\gamma)\mathcal{W}_{uu} & -(1-\gamma)\mathcal{W}_{uy} \\ -(1-\gamma)\mathcal{W}_{uy}^{\mathrm{T}} & \gamma \mathbf{K}_\eta^{-1} + (1-\gamma)\mathcal{W}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_1 \end{bmatrix} \\
&\qquad - \lambda_u(\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 - 1) - \lambda_y(\mathbf{y}_1^{\mathrm{T}}\mathbf{y}_1 - 1)
\end{aligned} \tag{3.6}$$

where λ_u and λ_y are the Lagrange multipliers. Then, $\partial\mathcal{L}/\partial\mathbf{u}_1 = \mathbf{0}$ and $\partial\mathcal{L}/\partial\mathbf{y}_1 = \mathbf{0}$ are reformulated as follows,

$$\begin{aligned}
&\begin{bmatrix} \gamma \mathbf{K}_\mu^{-1} + (1-\gamma)\mathcal{W}_{uu} & -(1-\gamma)\mathcal{W}_{uy} \\ -(1-\gamma)\mathcal{W}_{uy}^{\mathrm{T}} & \gamma \mathbf{K}_\eta^{-1} + (1-\gamma)\mathcal{W}_{yy} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_1 \end{bmatrix} \\
&\qquad = \begin{bmatrix} \lambda_u \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \lambda_y \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_1 \end{bmatrix}.
\end{aligned} \tag{3.7}$$

The determination of the vectors $\mathbf{u}_1$ and $\mathbf{y}_1$ and the constants λ_u and λ_y, which satisfy this equation, is called the multi-eigenvalue problem. The solution of the problem (3.1) corresponds to the vector associated with the minimum value of $\lambda_u + \lambda_y$. As a numerical solver of the MEP, the Horst-Jacobi method shown in Algorithm 4 can be applied.

Updating the latent system

Second, we update the system Σ using the obtained low-dimensional data $\mathbf{U}, \mathbf{Y}$. This problem is then defined as an optimization problem as follows:

$$\min_{\mathbf{G}(q)} f_2^t\big(\mathbf{U}, \mathbf{Y}, \mathbf{G}(q)\big). \tag{3.8}$$

The solution for the optimization problem (3.8) for $\mathbf{G}(q)$ is obtained entrywise. Regarding the system as ARX model, the least square solution of the fitting error to G_{ij} is obtained as follows:

$$\hat{\boldsymbol{\theta}}_{ij} = \left(\mathbf{\Psi}_{ij}^{\mathrm{T}}\mathbf{\Psi}_{ij}\right)^{-1}\mathbf{\Psi}_{ij}^{\mathrm{T}}\mathbf{y}_i, \tag{3.9}$$

where, with $y_i(t) = u_j(t) = 0$ for $t < 0$,

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{ij} &= \begin{bmatrix} \hat{a}_1^{(i,j)} & \cdots & \hat{a}_n^{(i,j)} & \hat{b}_0^{(i,j)} & \cdots & \hat{b}_n^{(i,j)} \end{bmatrix}^{\mathrm{T}}, \\
\boldsymbol{\Psi}_{ij} &= \begin{bmatrix} -\mathcal{Y}_i & \mathcal{U}_j \end{bmatrix}, \\
\mathcal{Y}_i &= \begin{bmatrix} y_i(-1) & y_i(-2) & \cdots & y_i(-n) \\ y_i(0) & y_i(1) & \cdots & y_i(1-n) \\ \vdots & \vdots & \ddots & \vdots \\ y_i(T-2) & y_i(T-3) & \cdots & y_i(T-n-1) \end{bmatrix}, \\
\mathcal{U}_j &= \begin{bmatrix} u_j(0) & u_j(-1) & \cdots & u_j(1-n) \\ u_j(1) & u_j(0) & \cdots & u_j(2-n) \\ \vdots & \vdots & \ddots & \vdots \\ u_j(T-1) & u_j(T-2) & \cdots & u_j(T-n) \end{bmatrix},
\end{aligned}$$

for $i = 1, \dots, d_y$ and $j = 1, \dots, d_u$. The optimal $\mathbf{G}(q)$ is obtained using the coefficients $\boldsymbol{\theta}_{ij}$ for all i and j.

Algorithm 1 shows the summarized algorithm of IOMLTF.

Convergence

Here, we give the following theorem that the value of f will be monotonically decreasing.

Theorem 1. *Let us utilize the following representations:* $\mathbf{u}^{(k)}, \mathbf{y}^{(k)}$ *and* $\mathbf{G}^{(k)}(q)$ *as the solutions at k-th iteration,* $\mathbf{W}_u^{(k)}$ *and* $\mathbf{W}_y^{(k)}$ *as* $\mathbf{W}_u$ *and* $\mathbf{W}_y$ *associated with* $\mathbf{G}^{(k)}(q)$ *respectively, and* $f^{(k)}$ *as the value of function at k-th iteration. Then* $f^{(k)}$ *converges to a local optimal when* $k \to \infty$.

Proof. Let us drop $\mathbf{K}_\mu, \mathbf{K}_\eta$ and γ from the input arguments of f, f_1, and f_2 for the sake of simplicity. The following two steps are executed at each iteration: 1) Obtain the updated low-dimensional representations $\mathbf{u}^{(k+1)}, \mathbf{y}^{(k+1)}$, 2) Estimate the transfer function $\mathbf{G}^{(k+1)}(q)$.

At Step 1, f is minimized with the fixed $\mathbf{G}^{(k)}(q)$. Then, f_2^t, which is the second term of f, is minimized at Step 2. Note that this operation (i.e. obtaining $\mathbf{G}^{(k+1)}(q)$ from $\mathbf{U}^{(k+1)}$ and $\mathbf{Y}^{(k+1)}$) does not effect the first term of f. As a result,

Algorithm 1: Input-Output Manifold Learning with Transfer Function Models

```
Input  : $\mathcal{D} = \{\boldsymbol{\mu}(t), \boldsymbol{\eta}(t)\}_{t=0}^{T-1}$,      // high dim. dataset
         $\mathbf{U}^{(0)}$, $\mathbf{Y}^{(0)}$,                                       // Initial low-dim. rep.
         $\mathbf{G}^{(0)}(q)$,                                                         // initial transfer function.
         $\gamma$                                                                        // weighting factor
         $h$                                                                             // prediction horizon
Output: $\hat{\mathbf{U}}$, $\hat{\mathbf{Y}}$, $\hat{\mathbf{G}}(q)$
Compute correlation matrices $\mathbf{K}_\mu$ and $\mathbf{K}_\eta$ from $\mathcal{D}$.
$i \leftarrow 0$
repeat
    Obtain $\{\mathbf{U}^{(i+1)}, \mathbf{Y}^{(i+1)}\}$ by solving the problem (3.1) with initial value $\mathbf{U}^{(i)}, \mathbf{Y}^{(i)}$ and $\mathbf{G}^{(i)}(q)$.
    Obtain $\mathbf{G}^{(i+1)}(q)$ by Eq. (3.9) with $\{\mathbf{U}^{(i+1)}, \mathbf{Y}^{(i+1)}\}$.
    $i \leftarrow i + 1$.
until convergence
$\hat{\mathbf{U}} \leftarrow \mathbf{U}^{(i)}$, $\hat{\mathbf{Y}} \leftarrow \mathbf{Y}^{(i)}$.
Obtain estimates of the transfer function model $\hat{\mathbf{G}}(q)$ by Eq. (3.9) with $\hat{\mathbf{U}}$ and $\hat{\mathbf{Y}}$.
```

the following inequality is satisfied.

$$\begin{aligned} & f(\mathbf{U}^{(k+1)}, \mathbf{Y}^{(k+1)}, \boldsymbol{G}^{(k+1)}(q)) \\ & \leq f(\mathbf{U}^{(k+1)}, \mathbf{Y}^{(k+1)}, \mathbf{G}^{(k)}(q)) \\ & \leq f(\mathbf{U}^{(k)}, \mathbf{Y}^{(k)}, \mathbf{G}^{(k)}(q)) \\ & \Leftrightarrow f^{(k+1)} \leq f^{(k)} \end{aligned} \tag{3.10}$$

Consequently, $f^{(k)}$ monotonically decreases. Since $f^{(k)} > 0$, it converges to a local optimal. As a result, the repeat of two steps converge f. □

3.4. Experiment

In this section, we evaluate the proposed method using two numerical simulations. First, the simulation settings are described in Sec. 3.4.1. Second, the parameter settings of the proposed method are shown in Sec. 3.4.2. Section 3.4.3 explains the comparison methods for showing the effectiveness of the proposed method, and the criterion in the evaluation is shown in Sec. 3.4.4. The results of each experiment are shown in Sec. 3.4.5.

3.4.1 Settings of experiments

Experiment 1 is designed such that the dimension of the high-dimensional input and output is controllable in order to validate the robustness of the proposed method for high dimensionality. Now, let us consider the SISO system, and change the dimension of the high-dimensional input and output from 2 to 5. This high-dimensional data are generated by the following procedure: the low-dimensional input $u(t)$ is generated by the value which follows the uniform distribution in $[-0.5\ 0.5]$, and the output y is obtained through the following transfer function,

$$G(q) = \frac{0.7 - 0.3q^{-1} + 0.2q^{-2}}{1 - 0.5q^{-1} - 0.1q^{-2}} \tag{3.11}$$

Then, the high-dimensional input $\boldsymbol{\mu}$ and output $\boldsymbol{\eta}$ are generated by the following procedure: Let us set the same dimension $d_\mu = d_\eta$. The j-th entry of $\boldsymbol{\mu}$ for $j = 1, \dots, d_\mu$ is represented as follows,

$$\mu_j(t) = u(t) \sin\left(u(t) + \frac{j-1}{d_u}\pi\right) + \epsilon(t),\ \ \epsilon(t) \sim \mathcal{N}(0, 0.05^2)$$

We apply the same procedure for $\boldsymbol{\eta}$. Figure 3.2 illustrates an example of two dimensional input and output.

Experiment 2 is designed with the image data, which is known as the effective data, for applying the manifold learning in order to validate the effectiveness for real data. In this experiment, the identification problem for the system shown in Fig. 3.3 is considered. In this experiment, the functions between u and $\boldsymbol{\mu}$, and between y and $\boldsymbol{\eta}$ are unknown. Nevertheless, this is a more realistic problem setting as compared to Experiment 1. First, the smooth angle sequence of the left-turntable $\{u(t)\}_{t=0}^{500-1}$, and the angle sequence of the right-turntable $\{y(t)\}_{t=0}^{500-1}$ through the transfer function (3.11) are determined. The image corresponding to each angle has been generated from COIL-100 [19] data-set, and it is used as the high-dimensional input and output ($d_\mu = d_\eta = 49,152$). Note that we

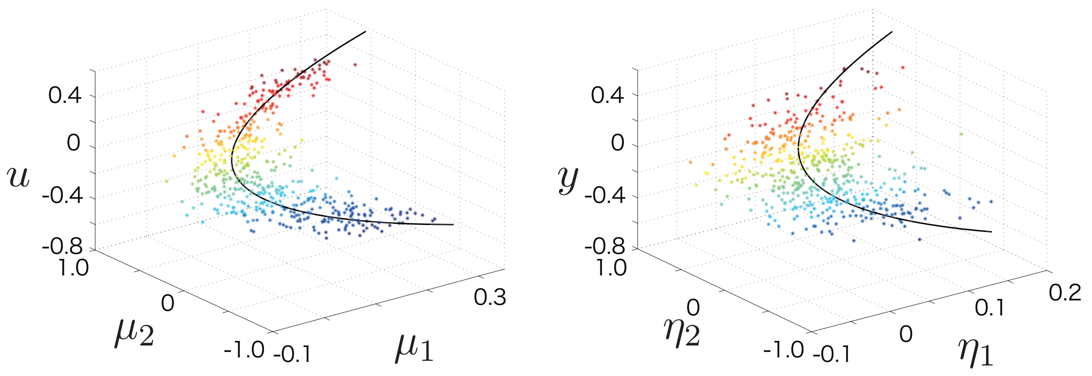


Figure 3.2. Input and output data used in experiment 1 (two dimensional case)

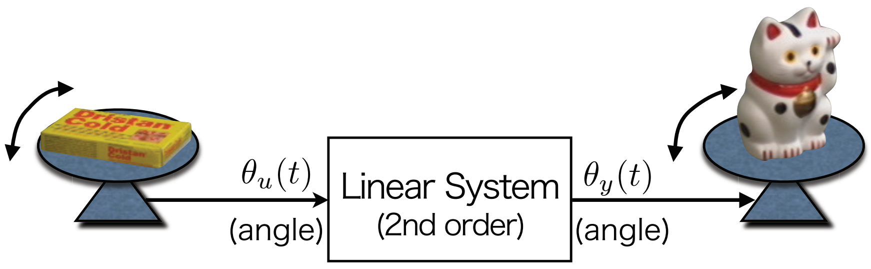


Figure 3.3. Problem setting using image data as input and output in experiment 2

have utilized kernel regression with the squared exponential kernel to generate the image from angle information since COIL-100 is associated with the image for each 5 degrees of rotation. It is expected that this image sequence will be lie on 1-dimensional manifold corresponding to the angle information. The image data for validation is newly generated in $u(t) \in [0, 90]$.

We also note that the system order $n = 2$ is assumed to be known.

3.4.2 Parameter settings for the proposed method

In Experiment 1, the number of the nearest neighbor in ISOMAP is set to 20, and the weighting factor γ is set to $\gamma = 0.1$ in a trial-and-error manner. We continue the iterative optimization $n_{\max} = 50$ times. Horst-Jacobi method [18] is used to solve MEP. The nonlinear regression in Step 6 is done with LS-SVM [20] whose

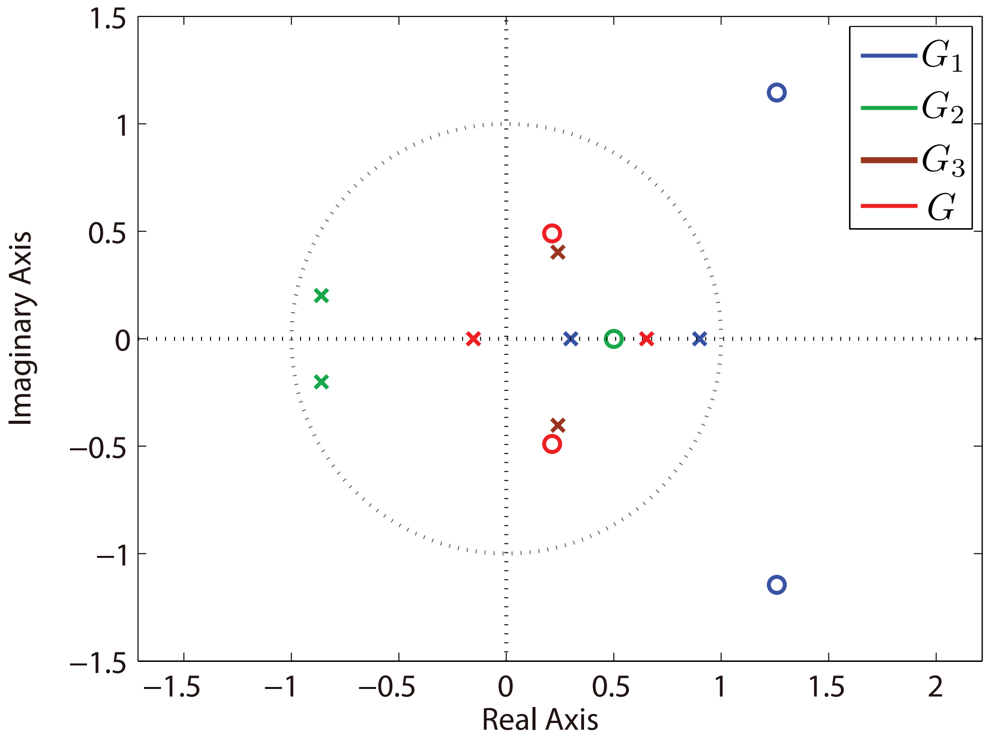


Figure 3.4. Pole-Zero map of G_1, G_2, G_3 and G. $\times$ represents the pole, and $\circ$ represents the zero.

hyperparameters are determined by the leave-one-out cross validation. The initial transfer function is selected from the following three transfer functions.

$$\begin{aligned} G_1 &= \frac{-0.29 + 0.73q^{-1} - 0.84q^{-2}}{1 - 1.2q^{-1} + 0.27q^{-2}} \\ G_2 &= \frac{-0.04q^{-1} + 0.02q^{-2}}{1 + 1.72q^{-1} + 0.78q^{-2}} \\ G_3 &= \frac{0.14q^{-2}}{1 - 0.48q^{-1} + 0.22q^{-2}} \end{aligned}$$

The pole-zero map of these transfer functions and the transfer function (3.11) is shown in Fig. 3.4.

In Experiment 2, the weighting factor γ is set to $\gamma = 0.01$ in a trial-and-error manner, and the maximum number of iterations $n_{\max}$ is set to 10. The nonlinear regression in Step 6 is done with the kernel regression with the squared exponential kernel, and the initial transfer function is selected as

$$G_{\text{init}} = \frac{1.0 - 1.0q^{-1} + 1.0q^{-2}}{1.0 + 0.05q^{-2}}.$$

The other settings are the same as those in Experiment 1.

3.4.3 Comparison methods

In Experiment 1, we utilize the following five methods (four for comparison, and our proposed method).

(a) Linear subspace system identification method (`4SID`)

(b) Nonlinear system identification method 1 (`NLHW`)

(c) Nonlinear system identification method 2 (`KCCA+LS-SVM,` [20])

(d) Applying the manifold learning method for the input and output separately, then identifying the system (`Isomap`). This is the same as the $\gamma = 1$ case in the proposed method)

(e) Proposed method (`Proposed`)

Method (a) is applied because the system to be identified is a linear system. Next, this identification problem can be deemed as the identification problem of a Hammerstein-Wiener system because the observed (high-dimensional) input and output can be regarded as the data through a nonlinear function; therefore, method (b) is applied. We also utilize method (c) that regards the system as a nonlinear dynamical system and reduce the dimensionality using the kernel canonical correlation analysis. Method (d) is applied to validate the penalty term which represents the fitting error to the dynamics. For methods (a) and (b), we use MATLAB System Identification Toolbox with default settings. In method (b), the nonlineality is estimated with a piecewise linear function. In method (c), the Gaussian kernel is utilized, and its hyperparameters are determined by 4-fold cross validation. The hyperparameters of the SVM are determined by the 10-fold cross validation.

In Experiment 2, Only method (d) is applied since methods (a), (b), and (c) are computationally complex.

3.4.4 Evaluation criteria

In Experiment 1, the fitting rate (FIT) of y with the filtered state with the Kalman filter is used for the evaluation*. Note that the Unscented Kalman Filter (UKF) is used because of the nonlinear dynamics. In Experiment 2, the correlation coefficient between the images is used for the evaluation.

3.4.5 Results and discussion

Experiment 1: with synthetic data

By using 500 data points for identification data and 500 data points for validation data, we apply five methods with the transfer function G_1 as the initial transfer function. The result for the varied dimensionality (average of 20 trials) is shown in Fig. 3.5.

Let us discuss Fig. 3.5. First, the FIT value decreases with high dimensional data in method (a). On the other hand, methods (b), (c), (d), and (e) keep the FIT value. However, the value of methods (b) and (c) is relatively small. Moreover, the FIT value in method (d) is larger than the FIT value in method (e). From these results, it is validated that the performance of the system is accumulated with the proposed method. Note that the significant difference between methods (d) and (e) is indicated by 2-sample t-test ($p < 0.05$).

Next, the fitting error to the linear system is shown in Table 3.1. Table 3.1 shows that the error of (d) is 1000 times larger than (e); therefore, it can be indicated that method (e) can find the suitable low-dimensional representation for the identification of linear dynamics. That is, the penalty term decreases the fitting error. The higher FIT value of (e) may be attributed to this, and

*FIT (%) is given as

$$\mathrm{FIT} = \left(1 - \frac{\sqrt{\sum_{t=0}^{N-1}(\hat{y}(t) - y(t))^2}}{\sqrt{\sum_{t=0}^{N-1}(y(t) - \bar{y})^2}}\right) \times 100$$

where $\hat{y}(t)$ is the actual data of $y(t)$ [21]. $\bar{y}$ represents the sample mean of $y(t)$. The FIT value varies from $-\infty$ to 100, however we utilize FIT $= 0$ if FIT < 0. For MIMO cases, the average of the FIT value is utilized as the FIT value.

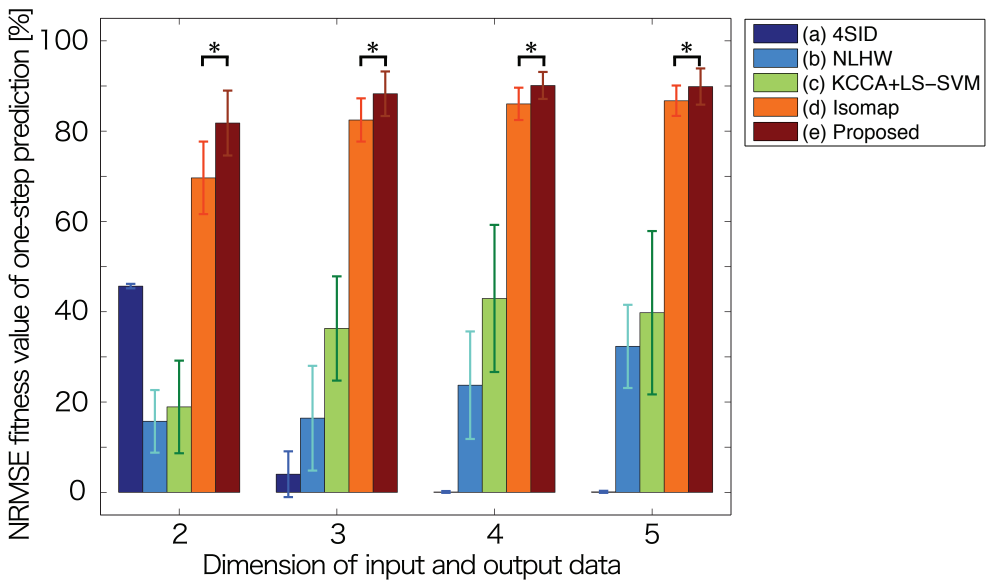


Figure 3.5. Normalized root mean square (NRMSE) fitness value in percentage vs. the dimension of input-output data

the penalty term with the fitting error ameliorates the linear transfer function estimation.

In addition, we confirm the effect of the initial transfer function. Using the 2-dimensional data as the high-dimensional input and output, we vary the initial transfer function in G_2, G_3, G. The mean and standard deviation of the FIT value is shown in Table 3.2. The two-sample t-test (p¿0.05) shows that a significant difference for G_1 is not observed. Consequently, the proposed method can achieve the accurate identification independently of the initial transfer function.

Experiment 2: with image data

The images for validation and the prediction are shown in Fig. 3.6. Also the fitting error to the linear system is shown in Table 3.3.

Let us discuss this result from the point of view of the penalty term. The value of (d) is smaller than (e), and it is more significant than Experiment 1 (Fig. 3.5). Moreover, Table 3.3 shows that the fitting error is significantly reduced by method (e). We suppose that this can be attributed to the fact that the higher dimensionality of the image deteriorates the performance of the manifold learning,

Table 3.1. Experiment 1: Identification error of $\hat{G}(q)$. This means the squared norm of $\breve{\boldsymbol{e}}$, $\|\breve{\boldsymbol{e}}\|^2$, which defined by Eq. (3.4). Numbers within brackets indicate standard deviations.

$m = p$	(d) Isomap	(e) Proposed
2	3.086e+00 (2.804e+00)	3.855e-03 (8.173e-04)
3	3.424e+00 (2.855e+00)	1.005e-03 (2.192e-04)
4	3.505e+00 (2.938e+00)	3.493e-03 (5.490e-05)
5	3.701e+00 (2.830e+00)	1.632e-04 (2.531e-05)

Table 3.2. Experiment1: Initial parameter of the transfer function vs. NRMSE fitness value in percentage. Numbers within brackets indicate standard deviations.

	G_1	G_2	G_3	G
FIT	81.7879	81.8390	81.5149	81.8645
	(7.1854)	(6.9840)	(6.7578)	(6.9236)

which is an unsupervised learning method. Therefore, the proposed method is valid for high dimensional data such as images.

3.5. Summary

In this chapter, we consider the system identification problem from high-dimensional input and output, and we propose the input-output manifold learning based on the manifold learning method considering the fitting error to the transfer function model. To show the effectiveness of the proposed method, we perform the system identification using synthetic data.

In general, the manifold is embedded to the high-dimensional data space. This can induce the interpretation that this system identification problem is a system identification problem of the Hammerstein-Wiener model, which is a block-oriented model with input and output with nonlinear function. Previously, many system identification methods for the Hammerstein-Wiener model [22, 23, 24] were proposed. However, these methods do not consider the high dimensionality, therefore an accurate model may not be obtained, as seen in Section 3.4.

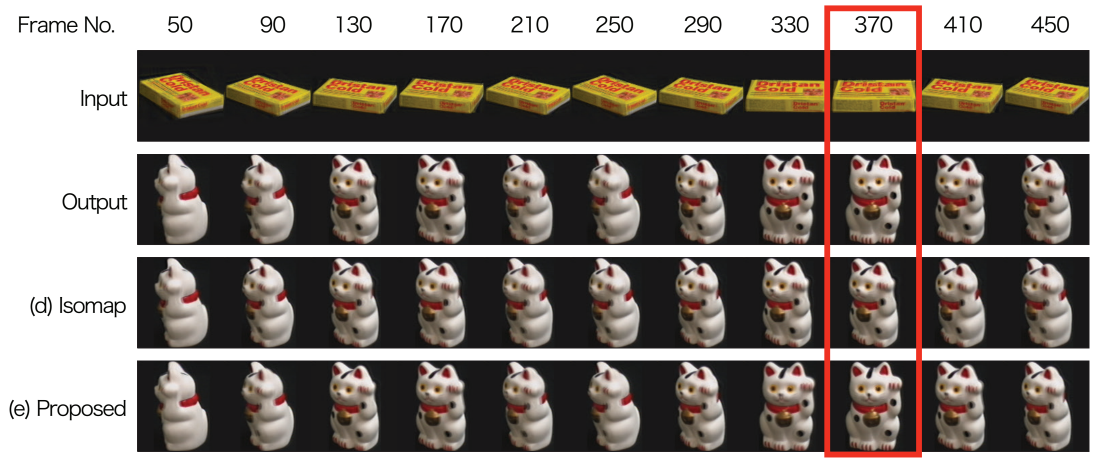


Figure 3.6. An example using image data for both the input and the output. From top to bottom, the image as the input for validation, the image as the output for validation, the one-step prediction result with Isomap, and the one-step prediction result with proposed input-output manifold learning. In particular at Frame No. 370, (e) makes a better prediction compared to (d). The correlation coefficient between the output and (d) is 0.8895, and between the output and (e) is 0.9949.

Table 3.3. Experiment 2: Identification error of $\hat{G}(q)$

	$\|\check{\boldsymbol{e}}\|^2$
(d) Isomap	8.0921
(e) Proposed	4.1134×10^{-12}

Next, let us describe the scope for our future works. First, we describe a theorem (Theorem 1) that shows that the error will decrease monotonically, and converge to a local optima. In addition, the solver for MEP used here (Horst-Jacobi method) or the quadratic programming problem (such as the interior-point method) is the local solver. Thus, obtaining the global solution is not clear as of now. Given that the global solver for the MEP has been recently proposed [25], application of these methods can be regarded as our future work.

We have assumed the transfer function model as the system representation. However, if the initial state of the system is non-zero when the high-dimensional data is obtained, it is difficult to obtain an accurate transfer function model. In the next chapter, we consider changing the system representation.

Chapter 4

Modeling of Non-relaxed System using High-dimensional Data

4.1. Introduction

In the previous chapter, we proposed input-output manifold learning with transfer function models (IOMLTF). Since the fitting error for the transfer function model can easily be reformulated to a quadratic form and model estimation can be accomplished element-wise, we have adopted the transfer function model as a system representation. However, the transfer function model has the following limitations:

1) It implicitly assumes that the system is initially relaxed (e.g., the initial state of the system is zero). That is, the transient response is difficult to capture.

2) A 'biased' estimation can be obtained if the model structure assumed is different from supposed one. [21, Sec. 8.3].

When we apply input-output manifold learning to several applications, these limitations may deteriorate the system identification performance. Therefore, in this chapter, we propose an alternative approach, i.e., the input-output manifold learning with state space models (IOMLSS), which can overcome the limitations of the previously proposed IOMLTF by introducing a state space model as the system representation. The state space model obviously allows representation of initially non-relaxed systems [26]. Moreover, sophisticated system identification methods such as N4SID [27], which can treat more complicated system models such as ARMAX models, can be utilized. Therefore, a better system identification performance can be expected.

The remainder of this chapter is organized as follows. In section 4.2, we explain our problem setting. Then, IOMLSS is described in section 4.3. A numerical simulation for validating IOMLSS is shown in section 4.4. The summary and future work are presented in section 4.5.

4.2. Problem setting

In this section, we explain the problem setting of IOMLSS.

Let us consider an unknown linear time-invariant discrete-time dynamical system Σ with the following input and output

$$\mathbf{u}(t) := [u_1(t), \ldots, u_{d_u}(t)]^{\mathrm{T}} \in \mathbb{R}^{d_u},$$
$$\mathbf{y}(t) := [y_1(t), \ldots, y_{d_y}(t)]^{\mathrm{T}} \in \mathbb{R}^{d_y},$$

which are referred to as the low-dimensional input and output of the system, respectively. The objective is to determine this relationship from the high-dimensional input $\boldsymbol{\mu}(t) \in \mathbb{R}^{d_\mu}(d_\mu > d_u)$ and output $\boldsymbol{\eta}(t) \in \mathbb{R}^{d_\eta}(d_\eta > d_y)$. Here, we assume that the dataset of the high-dimensional input and output from time $t = 0$ to $t = T-1$, $\mathcal{D} = \left\{\boldsymbol{\mu}(t), \boldsymbol{\eta}(t)\right\}_{t=0}^{T-1}$, is available, and the system order n and the dimensionality of the low-dimensional input d_u and output d_y are given. Figure 4.1 illustrates this problem setting.

4.3. Proposed method: input-output manifold learning with state space model

In this section, we describe the IOMLSS. The formulation of the optimization problem to determine the low dimensional representation with the state space model is shown in Sec. 4.3.1. The derivation of solution methods for the iterative optimization scheme is presented in Sec. 4.3.2.

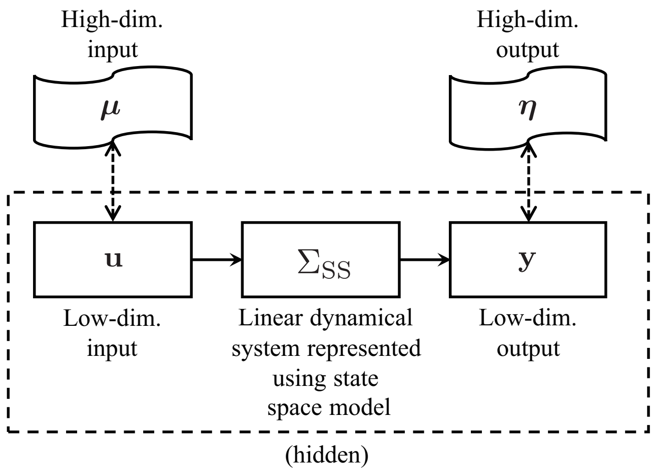


Figure 4.1. Problem setting of IOMLSS. The obtained data are the high-dimensional input $\boldsymbol{\mu}$ and output $\boldsymbol{\eta}$. It is assumed that the features regarded as low-dimensional input $\mathbf{u}$ and output $\mathbf{y}$ have a relationship of a linear dynamical system Σ_{SS}, represented using the state space model.

4.3.1 Formulation with state space model

As in the previous chapter, let us define the matrices consisting of data from $t = 0$ to $t = T - 1$ as follows:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^{\mathrm{T}}(0) \\ \vdots \\ \mathbf{u}^{\mathrm{T}}(T-1) \end{bmatrix} =: \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{d_u} \end{bmatrix},$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{\mathrm{T}}(0) \\ \vdots \\ \mathbf{y}^{\mathrm{T}}(T-1) \end{bmatrix} =: \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_{d_y} \end{bmatrix}.$$

The system Σ can be identified from $\mathbf{U}$ and $\mathbf{Y}$. However, all of $\mathbf{U}$, $\mathbf{Y}$ and Σ are unknown, and simultaneous estimation of all unknown variables is difficult. Hence, we have proposed an efficient algorithm called *input-output manifold learning*, as a solution method of two optimization problems. More concretely, assuming that the system Σ is given, we first solve the manifold learning problem reconstructed by adding the penalty term so that the low-dimensional data follow the dynamics[*]:

$$\begin{aligned} &\max_{\mathbf{U},\mathbf{Y}} f(\mathbf{U},\mathbf{Y},\Sigma_{\mathrm{SS}},\mathbf{K}_\mu,\mathbf{K}_\eta,\gamma) \\ &\text{s.t.}\, \mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_{d_u},\ \ \mathbf{Y}^{\mathrm{T}}\mathbf{Y} = \mathbf{I}_{d_y}. \end{aligned} \tag{4.1}$$

where

$$\begin{aligned} &f(\mathbf{U},\mathbf{Y},\Sigma_{\mathrm{SS}},\mathbf{K}_\mu,\mathbf{K}_\eta,\gamma) \\ &\quad := f_1(\mathbf{U},\mathbf{Y},\mathbf{K}_\mu,\mathbf{K}_\eta) - \gamma f_2\big(\mathbf{U},\mathbf{Y},\Sigma_{\mathrm{SS}}\big). \end{aligned} \tag{4.2}$$

The function $f_1(\mathbf{U},\mathbf{Y},\mathbf{K}_\mu,\mathbf{K}_\eta)$ with the constraints in equation (4.1) is used to determine the eigenvectors of $\mathbf{K}_\mu$ and $\mathbf{K}_\eta$,

$$f_1(\mathbf{U},\mathbf{Y},\mathbf{K}_\mu,\mathbf{K}_\eta) := \frac{1}{c_\mu}\mathrm{Tr}\big(\mathbf{U}^{\mathrm{T}}\mathbf{K}_\mu\mathbf{U}\big) + \frac{1}{c_\eta}\mathrm{Tr}\big(\mathbf{Y}^{\mathrm{T}}\mathbf{K}_\eta\mathbf{Y}\big),$$

which is the same as the original manifold learning method. Here, the matrices $\mathbf{K}_\mu, \mathbf{K}_\eta$ are defined by the gram matrix of Isomap for each, and the coefficients with $c_\mu := \lambda_{\max}(\mathbf{K}_\mu)$ and $c_\eta := \lambda_{\max}(\mathbf{K}_\eta)$ are multiplied for stable computation of numerical optimization. In addition, the function $f_2\big(\mathbf{U},\mathbf{Y},\Sigma_{\mathrm{SS}}\big)$ represents the fitting error for the system. The weighting factor $\gamma \geq 0$ is a free parameter to be manually tuned.

Let us consider a state space model as a system representation:

$$\Sigma_{\mathrm{SS}} : \begin{cases} \mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{v}(t) \end{cases}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector with the initial state $\mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^n$ and $\mathbf{w}(t)$ and $\mathbf{v}(t)$ are the system noise and observation noise respectively represented as follows,

$$p(\mathbf{w},\mathbf{v}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} \middle\| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{Q} & \mathbf{N} \\ \mathbf{N}^{\mathrm{T}} & \mathbf{R} \end{bmatrix}\right).$$

[*]Note that this formulation is slightly different from that in the IOMLTF. There is no problem with using the previous definition in Eq. (3.1). However, this formulation is numerically more stable in the optimization.

For the fitting error term, we utilize the following squared prediction error criterion of 4SID proposed in [28],

$$\begin{aligned} f_2(\mathbf{U}, \mathbf{Y}, \Sigma_{\mathrm{SS}}) &= f_2^s(\mathbf{U}, \mathbf{Y}, \mathbf{L}_w, \mathbf{H}_h^d) \\ &:= \left\| \mathbf{Y}_f - \begin{bmatrix} \mathbf{L}_w & \mathbf{H}_h^d \end{bmatrix} \begin{bmatrix} \mathbf{W}_p \\ \mathbf{U}_f \end{bmatrix} \right\|_{\mathrm{F}}^2, \end{aligned} \tag{4.3}$$

The matrices $\mathbf{Y}_*$ are parts of the following block Hankel matrices constructed from $\mathbf{y}$ expressed as follows:

$$\begin{aligned} \mathbf{Y}_{0|2h-1} &= \left[\begin{array}{cccc} \mathbf{y}(0) & \mathbf{y}(1) & \cdots & \mathbf{y}(T-2h) \\ \mathbf{y}(1) & \mathbf{y}(2) & \cdots & \mathbf{y}(T-2h+1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}(h-1) & \mathbf{y}(h) & \cdots & \mathbf{y}(T-h-1) \\ \hline \mathbf{y}(h) & \mathbf{y}(h+1) & \cdots & \mathbf{y}(T-h) \\ \mathbf{y}(h+1) & \mathbf{y}(h+2) & \cdots & \mathbf{y}(T-h+1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}(2h-1) & \mathbf{y}(2h) & \cdots & \mathbf{y}(T-1) \end{array} \right] \\ &=: \left[\begin{array}{c} \mathbf{Y}_p \\ \hline \mathbf{Y}_f \end{array} \right]. \end{aligned}$$

$\mathbf{U}_p$ and $\mathbf{U}_f$ are defined in the same way as $\mathbf{Y}_*$. In addition, $\mathbf{W}_p$ is defined as $\mathbf{W}_p := \begin{bmatrix} \mathbf{Y}_p \\ \mathbf{U}_p \end{bmatrix}$. The matrices $\mathbf{L}_w$ and $\mathbf{H}_h^d$ contain the information of the system Σ_{SS} (see Appendix C) and these are correspond to the transfer matrix $\mathbf{G}(q)$ in IOMLTF. An integer h is set to be larger than the system order n, i.e., $h > n$. See Appendix C for a rough overview of how this relationship is obtained.

Based on this representation, the IOMLSS is constructed as shown in Algorithm 2. In the following sections, detailed methods for obtaining the low-dimensional input and output, in order to update the latent system and to realize the system parameters are described.

Algorithm 2: Input-Output Manifold Learning with State Space Models

```
Input  : $\mathcal{D} = \{\boldsymbol{\mu}(t), \boldsymbol{\eta}(t)\}_{t=0}^{T-1}$,          // high dim. dataset
         $\mathbf{U}^{(0)}$, $\mathbf{Y}^{(0)}$,                                   // Initial low-dim.  rep.
         $\gamma$                                                  // weighting factor
         $h$                                                 // prediction horizon
         $n$                                                       // system order
Output: $\hat{\mathbf{U}}$, $\hat{\mathbf{Y}}$, $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$, $\hat{\mathbf{C}}$, $\hat{\mathbf{D}}$, $\hat{\mathbf{Q}}$, $\hat{\mathbf{R}}$, $\hat{\mathbf{N}}$
Compute correlation matrices $\mathbf{K}_\mu$ and $\mathbf{K}_\eta$ from $\mathcal{D}$.
Obtain $\{\mathbf{L}_w^{(0)}, \mathbf{H}_h^{d\,(0)}\}$ using Eq. (4.5) with $\{\mathbf{U}^{(0)}, \mathbf{Y}^{(0)}\}$.
$i \leftarrow 0$
repeat
    Obtain $\{\mathbf{U}^{(i+1)}, \mathbf{Y}^{(i+1)}\}$ by solving $\max_{\boldsymbol{\zeta}} f(\boldsymbol{\zeta}, \mathcal{K}, \mathcal{R}, \gamma)$ with initial value
    $\mathbf{U}^{(i)}, \mathbf{Y}^{(i)}$ and $\{\mathbf{L}_w^{(i)}, \mathbf{H}_h^{d\,(i)}\}$.
    Obtain $\{\mathbf{L}_w^{(i+1)}, \mathbf{H}_h^{d\,(i+1)}\}$ using Eq. (4.5) with $\{\mathbf{U}^{(i+1)}, \mathbf{Y}^{(i+1)}\}$.
    $i \leftarrow i + 1$.
until convergence
$\hat{\mathbf{U}} \leftarrow \mathbf{U}^{(i)}$, $\hat{\mathbf{Y}} \leftarrow \mathbf{Y}^{(i)}$.
Obtain estimates of system matrices $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$, $\hat{\mathbf{C}}$, $\hat{\mathbf{D}}$, $\hat{\mathbf{Q}}$, $\hat{\mathbf{R}}$, $\hat{\mathbf{N}}$ by Eqs. (4.6)
and (4.7) with $\hat{\mathbf{U}}$ and $\hat{\mathbf{Y}}$.
```

4.3.2 Solution methods

Obtaining low-dimensional input and output

Let us assume the matrices $\mathbf{L}_w$ and $\mathbf{H}_h^d$ are given. Now, the evaluation function (4.2) can be optimized with respect to $\mathbf{u}$ and $\mathbf{y}$. However, as the number of variables to be optimized is $(d_u + d_y)N$, it might be costly to obtain the derivative of the evaluation function (4.2) numerically. The objective is to determine the quadratic form of f because its analytic gradient can be easily obtained.

For f_1, we can obtain the quadratic form

$$\begin{aligned} & f_1(\mathbf{U}, \mathbf{Y}, \mathbf{K}_\mu, \mathbf{K}_\eta) \\ & = \begin{bmatrix} \mathrm{vec}(\mathbf{Y}) \\ \mathrm{vec}(\mathbf{U}) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathcal{K}_\eta & \mathbf{O}_{d_y N \times d_u N} \\ \mathbf{O}_{d_u N \times d_y N} & \mathcal{K}_\mu \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\mathbf{Y}) \\ \mathrm{vec}(\mathbf{U}) \end{bmatrix} \\ & = \boldsymbol{\zeta}^{\mathrm{T}} \mathcal{K} \boldsymbol{\zeta}, \end{aligned}$$

where

$$\mathcal{K}_\eta = \frac{1}{c_\eta} \mathbf{I}_{d_y} \otimes \mathbf{K}_\eta, \quad \mathcal{K}_\mu = \frac{1}{c_\mu} \mathbf{I}_{d_u} \otimes \mathbf{K}_\mu.$$

Next, to obtain a quadratic form of f_2^s, we utilize the following property of a Frobenius norm: $\|\mathbf{A}\|_{\mathrm{F}}^2 = \sum_i \|\mathbf{a}_i\|_2^2$ where $\mathbf{a}_i$ represents the i-th column vector of $\mathbf{A}$. Let us define the matrix inside of the Frobenius norm in equation (4.3) as $\mathbf{F}$,

$$\mathbf{F} := \mathbf{Y}_f - \begin{bmatrix} \mathbf{L}_w & \mathbf{H}_h^d \end{bmatrix} \begin{bmatrix} \mathbf{W}_p \\ \mathbf{U}_f \end{bmatrix}$$

and its i-th column vector as $\mathbf{f}_i$. The squared norm of $\mathbf{f}_i$ is as follows:

$$\|\mathbf{f}_i\|_2^2 = \left\| \mathbf{y}_{h+i|2h+i-1} - \begin{bmatrix} \mathbf{L}_w & \mathbf{H}_h^d \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \mathbf{y}_{i|h+i-1} \\ \mathbf{u}_{i|h+i-1} \end{bmatrix} \\ \mathbf{u}_{h+i|2h+i-1} \end{bmatrix} \right\|_2^2 . \tag{4.4}$$

Here, a notation $\mathbf{y}_{p|q} := \begin{bmatrix} \mathbf{y}^{\mathrm{T}}(p) & \cdots & \mathbf{y}^{\mathrm{T}}(q) \end{bmatrix}^{\mathrm{T}}$ is used. Let us split the matrix $\mathbf{L}_w$ into two matrices as

$$\mathbf{L}_w = \begin{bmatrix} \mathbf{L}_{wy} & \mathbf{L}_{wu} \end{bmatrix},$$

where $\mathbf{L}_{wy} \in \mathbb{R}^{d_y h \times d_y h}, \mathbf{L}_{wu} \in \mathbb{R}^{d_y h \times d_u h}$. Equation (4.4) can be reformulated as

$$\|\mathbf{f}_i\|_2^2 = \begin{bmatrix} \mathbf{y}_{i|2h+i-1} \\ \mathbf{u}_{i|2h+i-1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{R}_{yy} & -\mathbf{R}_{yu} \\ -\mathbf{R}_{yu}^{\mathrm{T}} & \mathbf{R}_{uu} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{i|2h+i-1} \\ \mathbf{u}_{i|2h+i-1} \end{bmatrix}$$

where

$$\mathbf{R}_{yy} = \begin{bmatrix} \mathbf{L}_{wy}^{\mathrm{T}}\mathbf{L}_{wy} & -\mathbf{L}_{wy}^{\mathrm{T}} \\ -\mathbf{L}_{wy} & \mathbf{I}_{hd_y} \end{bmatrix},$$
$$\mathbf{R}_{yu} = \begin{bmatrix} -\mathbf{L}_{wy}^{\mathrm{T}}\mathbf{L}_{wu} & -\mathbf{L}_{wy}^{\mathrm{T}}\mathbf{H}_h^d \\ -\mathbf{L}_{wu} & \mathbf{H}_h^{d\mathrm{T}} \end{bmatrix},$$
$$\mathbf{R}_{uu} = \begin{bmatrix} \mathbf{L}_{wu}^{\mathrm{T}}\mathbf{L}_{wu} & \mathbf{L}_{wu}^{\mathrm{T}}\mathbf{H}_h^d \\ \mathbf{H}_h^{d\mathrm{T}}\mathbf{L}_{wu} & \mathbf{H}_h^{d\mathrm{T}}\mathbf{H}_h^d \end{bmatrix}.$$

Using this result, f_2^s is reformulated into a quadratic form as

$$\begin{aligned} & f_2^s\big(\mathbf{U}, \mathbf{Y}, \mathbf{L}_w, \mathbf{H}_l^d\big) \\ &= \begin{bmatrix} \mathrm{vec}(\mathbf{Y}) \\ \mathrm{vec}(\mathbf{U}) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathcal{R}_{yy} & -\mathcal{R}_{yu} \\ -\mathcal{R}_{yu}^{\mathrm{T}} & \mathcal{R}_{uu} \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\mathbf{Y}) \\ \mathrm{vec}(\mathbf{U}) \end{bmatrix} \\ &= \boldsymbol{\zeta}^{\mathrm{T}}\mathcal{R}\boldsymbol{\zeta}, \end{aligned}$$

where

$$\begin{aligned} \mathcal{R}_{yy} &= \mathbf{P}_{T\times d_y}^{\mathrm{T}} \left(\sum_{i=0}^{T-2h} \mathbf{R}_{yy}^{(i)} \right) \mathbf{P}_{T\times d_y}, \\ \mathbf{R}_{yy}^{(i)} &= \mathrm{diag}\big\{ \mathbf{O}_{(i-1)d_y\times(i-1)d_y}, \mathbf{R}_{yy}, \mathbf{O}_{(N-2h-i)d_y\times(N-2h-i)d_y} \big\}, \\ \mathcal{R}_{yu} &= \mathbf{P}_{T\times d_y}^{\mathrm{T}} \left(\sum_{i=0}^{T-2h} \mathbf{R}_{yu}^{(i)} \right) \mathbf{P}_{T\times d_u}, \\ \mathbf{R}_{yu}^{(i)} &= \mathrm{diag}\big\{ \mathbf{O}_{(i-1)d_y\times(i-1)d_u}, \mathbf{R}_{yu}, \mathbf{O}_{(N-2h-i)d_y\times(N-2h-i)d_u} \big\}, \\ \mathcal{R}_{uu} &= \mathbf{P}_{T\times d_u}^{\mathrm{T}} \left(\sum_{i=0}^{T-2h} \mathbf{R}_{uu}^{(i)} \right) \mathbf{P}_{T\times d_u}, \\ \mathbf{R}_{uu}^{(i)} &= \mathrm{diag}\big\{ \mathbf{O}_{(i-1)d_u\times(i-1)d_u}, \mathbf{R}_{uu}, \mathbf{O}_{(N-2h-i)d_u\times(N-2h-i)d_u} \big\}, \end{aligned}$$

and $\mathbf{P}_{p\times q}$ is a $pq \times pq$ commutation matrix represented as

$$\mathbf{P}_{p\times q} = \sum_{i=1}^{p}\sum_{j=1}^{q} \mathbf{E}_{ij}^{(p\times q)} \otimes \mathbf{E}_{ji}^{(q\times p)},$$

where $\mathbf{E}_{ij}^{(p\times q)}$ is a matrix unit that is a $p \times q$ matrix whose (i, j) entry is 1 and the rest are 0. Finally, the quadratic form of the evaluation function and its gradient are obtained as follows

$$f(\boldsymbol{\zeta}, \mathcal{K}, \mathcal{R}, \gamma) = \boldsymbol{\zeta}^{\mathrm{T}} \mathcal{M} \boldsymbol{\zeta}, \quad \frac{\partial f}{\partial \boldsymbol{\zeta}} = 2\mathcal{M}\boldsymbol{\zeta},$$

where $\mathcal{M} = \mathcal{K} - \gamma\mathcal{R}$. The solution of the optimization problem (4.1) can be obtained using a solver for quadratic programming with quadratic equality constraints such as SQP or an interior-point algorithm using this gradient.

Updating the latent system

In this section, we describe how to obtain the matrices $\mathbf{L}_w$ and $\mathbf{H}_h^d$, which minimize f_2^s for obtained $\mathbf{U}$ and $\mathbf{Y}$. In the IOMLTF, the solution is easily obtained using least squares estimation. Even in the IOMLSS, it seems that the solution might be easily obtained as

$$\left[\begin{array}{cc} \mathbf{L}_w & \mathbf{H}_h^d \end{array} \right] = \mathbf{Y}_f \left[\begin{array}{c} \mathbf{W}_p \\ \mathbf{U}_f \end{array} \right]^{\dagger}.$$

where † stands for the pseudo inverse $\mathcal{X}^{\dagger} = \mathcal{X}^{\mathrm{T}}(\mathcal{X}\mathcal{X}^{\mathrm{T}})^{-1}$. However, even $\mathbf{H}_h^d$ is a block lower triangular matrix (see Appendix C), this simple solution method does not have this structure. For a single-input single-output case, the solution that holds the structure is shown in [29, Sec. 5.2] as an example. However, the multi-input multi-output case is not shown explicitly. We here describe how to obtain the appropriate solution.

First, we consider the vectorized form of equation (C.3) as follows:

$$\underbrace{\mathrm{vec}(\mathbf{Y}_f)}_{\boldsymbol{\Upsilon}} = \underbrace{\Big(\left[\begin{array}{cc} \mathbf{W}_p^{\mathrm{T}} & \mathbf{U}_f^{\mathrm{T}} \end{array} \right] \otimes \mathbf{I}_{hd_y} \Big)}_{\boldsymbol{\Xi}} \mathrm{vec}\Big(\left[\begin{array}{cc} \mathbf{L}_w & \mathbf{H}_h^d \end{array} \right] \Big),$$

and define a matrix $\mathbf{H}$ constructed with the Markov parameters as:

$$\mathbf{H} = \left[\begin{array}{cccc} \mathbf{H}_0^{\mathrm{T}} & \mathbf{H}_1^{\mathrm{T}} & \cdots & \mathbf{H}_{h-1}^{\mathrm{T}} \end{array} \right]^{\mathrm{T}}.$$

The relationship between $\mathrm{vec}(\mathbf{H}_h^d)$ and $\mathbf{H}$ can be easily determined as

$$\mathrm{vec}(\mathbf{H}_h^d) = \mathbf{N}\mathrm{vec}(\mathbf{H})$$

where $\mathbf{N}$ is defined as

$$\mathbf{N} = \begin{bmatrix} \mathbf{I}_{d_u} \otimes \mathbf{N}_1 \\ \mathbf{I}_{d_u} \otimes \mathbf{N}_2 \\ \vdots \\ \mathbf{I}_{d_u} \otimes \mathbf{N}_h \end{bmatrix},$$

$$\mathbf{N}_i = \begin{bmatrix} \mathbf{O}_{(i-1)d_y \times (h-i+1)d_y} & \mathbf{O}_{(i-1)d_y \times (i-1)d_y} \\ \mathbf{I}_{(h-i+1)d_y} & \mathbf{O}_{(h-i+1)d_y \times (i-1)d_y} \end{bmatrix}.$$

Denoting

$$\begin{aligned} \operatorname{vec}\Big(\begin{bmatrix} \mathbf{L}_w & \mathbf{H}_h^d \end{bmatrix}\Big) &= \operatorname{diag}\Big\{\mathbf{I}_{(d_u+d_y)d_y h^2}, \mathbf{N}\Big\} \begin{bmatrix} \operatorname{vec}(\mathbf{L}_w) \\ \operatorname{vec}(\mathbf{H}) \end{bmatrix} \\ &= \mathcal{N}\boldsymbol{\zeta}, \end{aligned}$$

and solving the optimization problem

$$\min_{\boldsymbol{\zeta}} \Big\|\boldsymbol{\Upsilon} - \boldsymbol{\Xi}\mathcal{N}\boldsymbol{\zeta}\Big\|_2^2$$

which is equivalent to the minimization problem of f_2^s w.r.t. $\mathbf{L}_w$ and $\mathbf{H}_h^d$, the vector $\operatorname{vec}\Big(\begin{bmatrix} \mathbf{L}_w & \mathbf{H}_h^d \end{bmatrix}\Big)$ yields

$$\operatorname{vec}\Big(\begin{bmatrix} \mathbf{L}_w & \mathbf{H}_h^d \end{bmatrix}\Big) = \mathcal{N}\left(\mathcal{N}^{\mathrm{T}}\boldsymbol{\Xi}^{\mathrm{T}}\boldsymbol{\Xi}\mathcal{N}\right)^{-1}\mathcal{N}^{\mathrm{T}}\boldsymbol{\Xi}^{\mathrm{T}}\boldsymbol{\Upsilon}. \tag{4.5}$$

The updated matrices are obtained by reshaping this vector.

Estimation of the system parameters

After repeating the two optimization problems, the system parameters of Σ are estimated using the following algorithm based on [27]. First, we compute the SVD of $\mathbf{L}_w\mathbf{W}_p$:

$$\begin{aligned} \mathbf{L}_w\mathbf{W}_p &= \boldsymbol{\Gamma}_h\hat{\mathbf{X}}_h \quad \text{(from Eqs. (C.1) and (C.2))} \\ &= \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^{\mathrm{T}} \\ \mathbf{V}_2^{\mathrm{T}} \end{bmatrix}, \end{aligned}$$

where $\mathbf{S}_1$ is the $n \times n$ diagonal matrix. Using the components, the state sequence $\hat{\mathbf{X}}_h$ is estimated as:

$$\hat{\mathbf{X}}_h := \left[\begin{array}{cccc} \hat{\mathbf{x}}(h) & \hat{\mathbf{x}}(h+1) & \cdots & \hat{\mathbf{x}}(T-h) \end{array} \right] = \mathbf{S}_1^{1/2}\mathbf{V}_1^{\mathrm{T}}$$

The system parameters $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are obtained by solving the following least squares problem:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}} \|\boldsymbol{\epsilon}\|_{\mathrm{F}}^2, \quad \boldsymbol{\epsilon} := \left[\begin{array}{c} \mathbf{X}_h^p \\ \mathbf{Y}_h \end{array} \right] - \left[\begin{array}{cc} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{array} \right] \left[\begin{array}{c} \mathbf{X}_h^c \\ \mathbf{U}_h \end{array} \right],$$

whose solutions is obtained by

$$\left[\begin{array}{cc} \hat{\mathbf{A}} & \hat{\mathbf{B}} \\ \hat{\mathbf{C}} & \hat{\mathbf{D}} \end{array} \right] = \left[\begin{array}{c} \mathbf{X}_h^p \\ \mathbf{Y}_h \end{array} \right] \left[\begin{array}{c} \mathbf{X}_h^c \\ \mathbf{U}_h \end{array} \right]^{\dagger} \tag{4.6}$$

where

$$\begin{aligned} \hat{\mathbf{X}}_h^p &:= \left[\begin{array}{cccc} \hat{\mathbf{x}}(h+1) & \hat{\mathbf{x}}(h+2) & \cdots & \hat{\mathbf{x}}(T-h) \end{array} \right], \\ \hat{\mathbf{X}}_h^c &:= \left[\begin{array}{cccc} \hat{\mathbf{x}}(h) & \hat{\mathbf{x}}(h+1) & \cdots & \hat{\mathbf{x}}(T-h-1) \end{array} \right], \\ \mathbf{Y}_h &:= \left[\begin{array}{cccc} \mathbf{y}(h) & \mathbf{y}(h+1) & \cdots & \mathbf{y}(T-h-1) \end{array} \right], \\ \mathbf{U}_h &:= \left[\begin{array}{cccc} \mathbf{u}(h) & \mathbf{u}(h+1) & \cdots & \mathbf{u}(T-h-1) \end{array} \right]. \end{aligned}$$

The estimations of each covariance matrix of $\mathbf{w}$ and $\mathbf{v}$ are obtained using $\boldsymbol{\epsilon}_{\mathrm{opt}} = \arg\min\|\boldsymbol{\epsilon}\|_{\mathrm{F}}^2$,

$$\left[\begin{array}{cc} \hat{\mathbf{Q}} & \hat{\mathbf{N}} \\ \hat{\mathbf{N}}^{\mathrm{T}} & \hat{\mathbf{R}} \end{array} \right] = \frac{1}{N-h-2}\boldsymbol{\epsilon}_{\mathrm{opt}}\boldsymbol{\epsilon}_{\mathrm{opt}}^{\mathrm{T}}. \tag{4.7}$$

The difference in the algorithms for the IOMLTF and IOMLSS is summarized in Fig. 4.2. The main difference is in the parameters to be estimated after obtaining the low-dimensional input and output data.

4.4. Experiment

In this section, we illustrate the effectiveness of our proposed method through simulations with synthetic data. We compare the following three methods: 1) ISOMAP (with the system identified with low-dimensional data obtained by Isomap (same as the initial values of IOML)), 2) IOMLTF [30], and 3) IOMLSS (proposed).

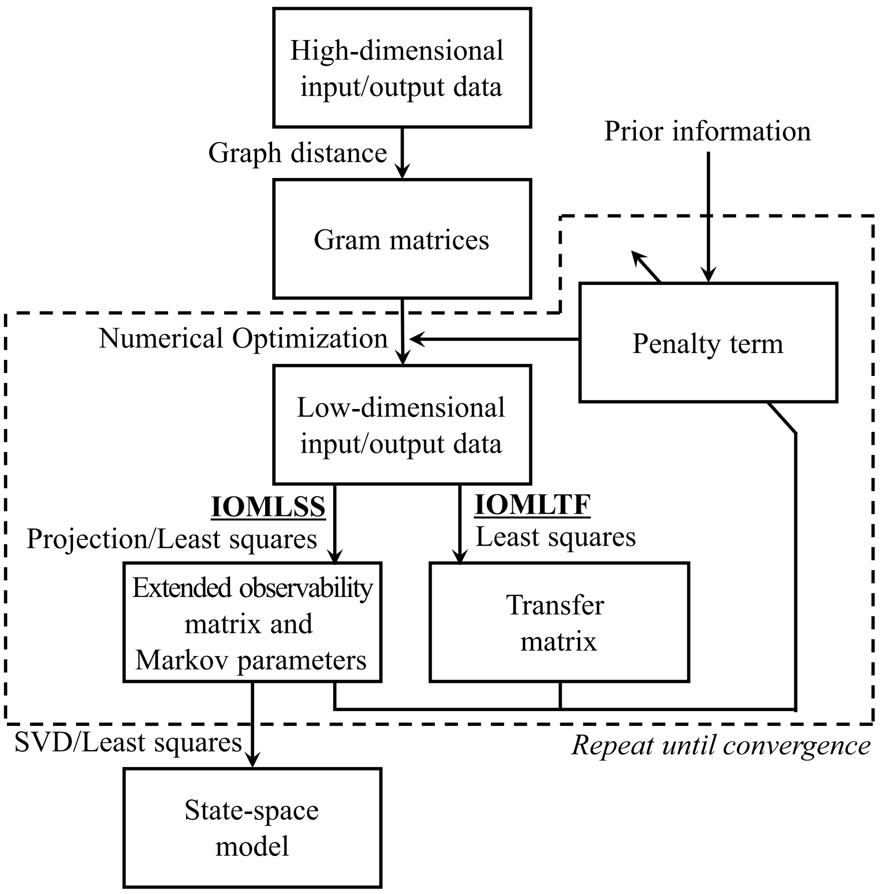


Figure 4.2. Algorithmic flow of IOMLSS and IOMLTF. The difference between the two is how the penalty term is updated. IOMLSS utilizes an extended observability matrix and Markov parameters, while IOMLTF utilizes the transfer function model.

4.4.1 Simulation settings

Training dataset

Let us assume the following 3rd order system Σ_1:

$$\Sigma_1 : \begin{cases} \mathbf{x}(t+1) = \mathbf{A}_1\mathbf{x}(t) + \mathbf{B}_1 u(t) + \mathbf{w}(t) \\ \qquad y(t) = \mathbf{C}_1\mathbf{x}(t) + v(t) \end{cases}$$

where

$$\begin{aligned}
\mathbf{A}_1 &= \begin{bmatrix} 0.2338 & 0.2869 & -0.1383 \\ 0.2869 & 0.3525 & 0.2786 \\ -0.1383 & 0.2786 & 0.3917 \end{bmatrix}, \\
\mathbf{B}_1 &= \begin{bmatrix} 0.7007 & 1.424 & 0 \end{bmatrix}^{\mathrm{T}}, \\
\mathbf{C}_1 &= \begin{bmatrix} 0 & 0.629 & -2.427 \end{bmatrix},
\end{aligned}$$

and $\mathbb{E}\{\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}(t)\} = 0.01\mathbf{I}$, $\mathbb{E}\{v^2(t)\} = 0.01$, and $\mathbb{E}\{\mathbf{w}(t)v(t)\} = \mathbf{0}$.

As the high-dimensional input and output, we consider two dimensional vectors obtained through the following nonlinear functions:

$$\begin{aligned}
\boldsymbol{\mu}(t) &= \begin{bmatrix} u(t) & \exp\left(-\frac{1}{2}u^2(t)\right) \end{bmatrix}^{\mathrm{T}} + \boldsymbol{\nu}_\mu(t), \\
\boldsymbol{\eta}(t) &= \begin{bmatrix} y(t) & \exp\left(-\frac{1}{2}y^2(t)\right) \end{bmatrix}^{\mathrm{T}} + \boldsymbol{\nu}_\eta(t),
\end{aligned}$$

where $\boldsymbol{\nu}_\mu(t) \sim \mathcal{N}(\mathbf{0}, 0.09\mathbf{I})$ and $\boldsymbol{\nu}_\eta(t) \sim \mathcal{N}(\mathbf{0}, 0.09\mathbf{I})$ are additional noises. The high-dimensional dataset $\mathcal{D} = \{\boldsymbol{\mu}(t), \boldsymbol{\eta}(t)\}_{t=0}^{T-1}$ with $T = 1000$ data point is generated with the low-dimensional input, which follows a normal distribution $u(t) \sim \mathcal{N}(0, 1)$. We consider two conditions:

1) a non-zero initial state $\mathbf{x}(0) = \mathbf{x}_0$ where
   $\mathbf{x}_0 = \begin{bmatrix} -0.1117 & -0.0446 & 0.0264 \end{bmatrix}^{\mathrm{T}}$
2) a zero initial state $\mathbf{x}(0) = \mathbf{0}$

Following the procedure, we prepare five dataset $\mathcal{D}_1, \ldots, \mathcal{D}_5$ for each condition because the low-dimensional input $\mathbf{u}$ and noises are random.

Evaluation criteria

In the evaluation, we focus on the output error with the filtered state estimated by the steady state Kalman filter, as used in the evaluation of IOMLTF [30]. First, two nonlinear functions, $\boldsymbol{\mu}(t) \to u(t)$ and $\boldsymbol{\eta}(t) \to y(t)$, are obtained with ϵ-SVR implemented by LIBSVM [31]. The hyperparameters in ϵ-SVR are optimized with 5-fold cross validation. Then, the new dataset $\mathcal{D}_t$ with $N_t = 500$ samples for the test is generated with the same conditions as those when $\mathcal{D}$ is generated. Using the SVRs, the low-dimensional input and output for the test data are computed. Then, the state $\mathbf{x}(t)$ is estimated using the steady state Kalman filter, and the difference between the low-dimensional output $\hat{y}(t)$ with the estimated state $\hat{\mathbf{x}}(t)$ and the actual observation $y(t)$ is evaluated based on an FIT value defined as follows:

$$\mathrm{FIT} = \max\left(0, 1 - \frac{\sum_{\tau=0}^{N_t-1}(y(\tau) - \hat{y}(\tau))^2}{\sum_{\tau=0}^{N_t-1}(y(\tau) - \bar{y})^2}\right) \times 100[\%],$$

where $\bar{y} = \frac{1}{N_t}\sum_{\tau=0}^{N_t-1} y(\tau)$. If the value is large, it is possible that the obtained low-dimensional input and output are more adapted to the linear system. In the design of the Kalman filter, the covariance matrices of the noises are given by Eq. (4.7) for IOMLSS and ISOMAP. As the matrices for IOMLTF are not given, the matrices obtained for IOMLSS are utilized for IOMLTF.

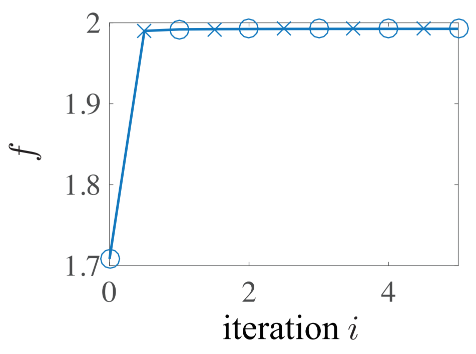
User parameter and initial value setting

The prediction horizon is set to $h = 20$. To set the weighting factor γ, we divide the dataset $\mathcal{D}_i$ into the training data (first 800 points) and the validation data (remaining 200 points). The weighting factor γ is selected from a set $\{0.01, 0.05, 0.1, 0.5, 1\}$ so that the FIT value of the validation data is maximum. The iterative optimization in IOMLTF and IOMLSS is repeated for 1000 iterations. The system order $n = 3$ is manually chosen.

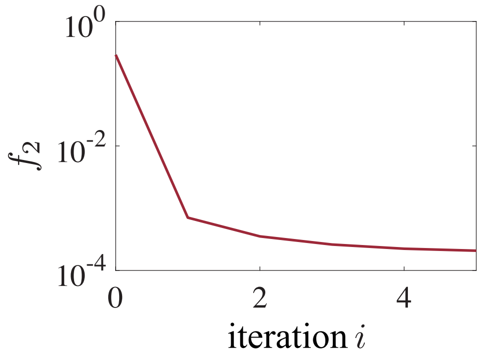
The initial values $\mathbf{U}^{(0)}, \mathbf{Y}^{(0)}$ are set as the solutions of Isomap [6], and the matrices $\mathbf{L}_w^{(0)}, \mathbf{H}_h^{d(0)}$ are obtained using $\mathbf{U}^{(0)}$ and $\mathbf{Y}^{(0)}$ with Eq. (4.5).

4.4.2 Results

First, we show the learning result of IOMLSS. Figure 4.3(a) shows the value of the evaluation function f for each iteration in IOMLSS. Although the value at the 0th iteration is small because the fitting to the linear system is worse, as shown in Fig. 4.3(b), the evaluation function f monotonically increases by repeating the optimizations because the fitting error term f_2 monotonically decreases. The effectiveness of the iterative optimization is confirmed by this result.



(a) Evaluation function f



(b) Fitting error f_2

Figure 4.3. Values of the evaluation function and the fitting error term in iterative optimization. In (a), $\circ$ denotes the value after the optimization of f_2, $f(\mathbf{U}^{(i)}, \mathbf{Y}^{(i)}, \mathbf{L}_w^{(i)}, \mathbf{H}_h^{d\,(i)})$, and $\times$ shows the value after the optimization of f, $f(\mathbf{U}^{(i+1)}, \mathbf{Y}^{(i+1)}, \mathbf{L}_w^{(i)}, \mathbf{H}_h^{d\,(i)})$. The value of f monotonically increases in each iteration as the value of f_2 monotonically decreases.

Figure 4.4 shows a comparison of the average of the FIT values. In the figure, the results of IOMLSS are better than those of IOMLTF and ISOMAP. The actual output and estimated outputs in test are shown in Fig. 4.5. These results indicate that IOMLSS can be used to obtain more suitable low-dimensional representations and the system for non-zero initial state, $\mathbf{x}(0) = \mathbf{x}_0$. In addition, as opposed to this case, there is a smaller difference between IOMLSS and IOMLTF for a zero initial state, $\mathbf{x}(0) = \mathbf{0}$. Nevertheless, it is confirmed that there is a significant difference between IOMLSS and IOMLTF validated by two-sample t-tests ($\mathrm{p} < 0.05$). We suppose that this is caused by the model difference because IOMLTF uses the ARX model and the system Σ_1 and IOMLSS uses ARMAX. This difference may induce a 'biased' estimation [21, Sec. 8.3] which may deteriorate the prediction performance. In other words, this result shows that IOMLSS can be used to treat a more complicated noise model than IOMLTF. Consequently, the effectiveness of the proposed method is confirmed.

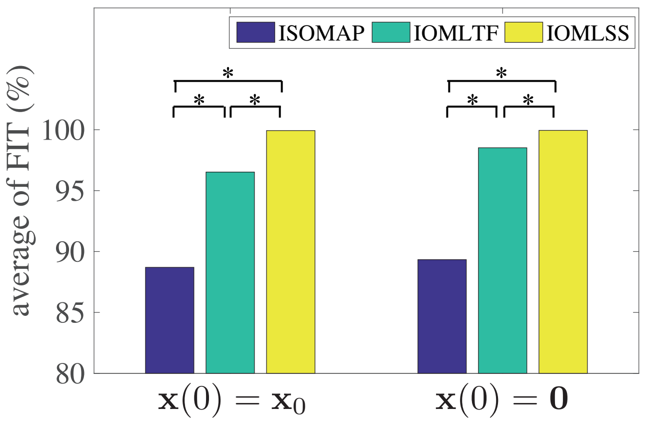


Figure 4.4. Average value of FIT value for test data. The y-axis shows the average of FIT value for five datasets for each condition. IOMLTF and IOMLSS are shown to make better predictions than ISOMAP. However, IOMLSS performs better than IOMLTF for a case with a non-zero initial state case (IOMLTF: 96.5%, IOMLSS: 99.9%), which is validated by two-sample t-tests ($p < 0.05$). The difference is smaller for a case with a zero initial state case (IOMLTF: 98.5%, IOMLSS: 99.9%).

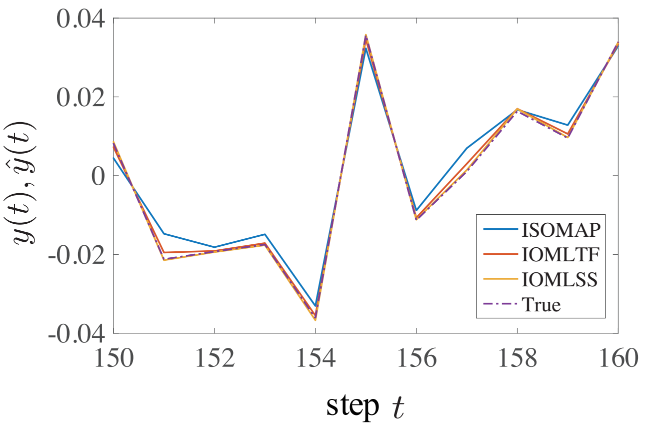


Figure 4.5. Estimated output sequence in the test with non-zero initial state case from $t = 150$ to $t = 160$. The output of IOMLSS is close to the true output y.

4.5. Summary

In this paper, we extend input-output manifold learning to use the state space model for the system representation. This extension allows us to consider a non-zero initial state and a more complicated model structure. The simulation result demonstrates that the extended method results in better low-dimensional representations if the initial state of the obtained dataset is non-zero.

By using the state space model for system representation, the subspace identification method can be implemented. We determine the system order n manually in the experiment. However, the system order can be determined from the identification data using this method. In addition, the number of parameters to be identified for the MIMO system can be reduced with the state space model. It is deemed that the performance will be better for MIMO system cases, However, IOMLTF can be a potential choice for certain systems because the number of parameters to be identified is less than that for the IOMLSS in specific cases.

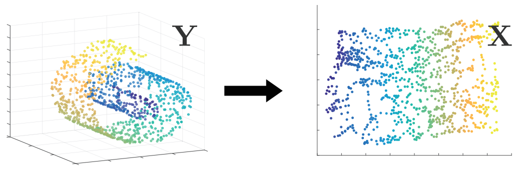
In future, we will consider the use of nonlinear dynamical systems. In addition, we will also consider applying this method to the actual data in future.

Chapter 5

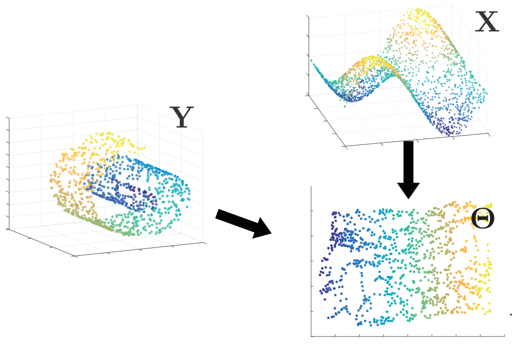
Partial Manifold Learning for Two Factor Models and Its Application to Active Robot Perception on Learned Manifolds

5.1. Introduction

In this section, a manifold learning method with a two-factor model is considered. In previous studies, the manifold learning methods for an one-factor model are proposed such as PCA [3], Isomap [6], and GP-LVMs [2]. The generative model considered in such a study is one factor models, which shows that the observation is explained by one factor, namely $\mathbf{y} = \mathbf{f}(\mathbf{X})$ where $\mathbf{y}$ is the observation, and the $\mathbf{X}$ is the latent factor (Fig. 5.1(a)). In this section, we consider models with two-factors, namely $\mathbf{y} = \mathbf{g}(\mathbf{x}, \boldsymbol{\theta})$. The two (or more) factor model is capable capturing variations in the data which is described in the problem of style-content separation. Such a model is applied to spoken vowel classification [32], and prediction of human motions [33]. In this study, we assume that one factor $\mathbf{x}$ is known (Fig. 5.1(b)). Obtaining both factor is considered in the previous studies (e.g. [32, 33]). However, the method can not be directly applied to the problem to be considered in this study. The situation would be happen in the sensor



(a) Single-factor generative model



(b) Two-factor generative model whose one factor is known and the other is unknown

Figure 5.1. Problem setting comparison. (a) Find $\mathbf{X}$ for given $\mathbf{Y}$. (b) Fing $\mathbf{\Theta}$ for given $\mathbf{Y}$ and $\mathbf{X}$.

model construction in the object recognition tasks (see Sec. 5.3). To cope with the problem, we propose a partial manifold learning method based on Gaussian Process Latent Variable Models (GP-LVMs) [2], which allows us to consider the effect of a known factor to the observation and other factor.

5.1.1 Layout of this chapter

Section 5.2 describes proposed manifold learning method, and its connection to object recognition problem is shown in section 5.3. Section 5.3 also describes the information maximization control, which is the collateral proposed method for exploratory action planning in object recognition task. The performance validation of object manifold learning and information maximization control is shown in section 5.4 and section 5.5, respectively. This chapter is summarized in section 5.6 with discussions.

5.2. Proposed method

We now start to model the relationship among the observation, known factor and unknown factor. As described above, the following nonlinear observation function

$\mathbf{g}$ is assumed:

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\epsilon}, \tag{5.1}$$

where $\mathbf{y}$ is the observation, $\mathbf{x}$ is the known factor, and $\boldsymbol{\theta}$ is the unknown factor.

5.2.1 Observation modeling using GPs

Firstly, the nonlinear function $\mathbf{g}$ is modeled using GPs. The observation model (5.1) is supposed to be modeled using Gaussian Process Regression [34] for each dimension of $\mathbf{y}$: $y_a = g_a(\mathbf{x}, \boldsymbol{\theta}) + \epsilon_a$ for $a = 1, 2, \ldots, d_y$ as follows:

$$g_a(\mathbf{z}) \sim \mathcal{GP}(0, k_a(\mathbf{z}, \mathbf{z}')), \tag{5.2}$$

where $k_a(\mathbf{z}, \mathbf{z}')$ is the kernel function. $\mathbf{z} = \left[\mathbf{x}^{\mathrm{T}}, \boldsymbol{\theta}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{d_z}$ where $d_z = d_x + d_\theta$ is defined for simple representation.

For given N-sample training data set $\tilde{\mathcal{D}} = \{\mathbf{x}^{(i)}, \boldsymbol{\theta}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ (Note that $\boldsymbol{\theta}$ is not given at this moment), the predictive distribution of y_a is given as a Gaussian distribution:

$$\begin{aligned}&p(y_a|\mathbf{x}, \boldsymbol{\theta}, \mathbf{X}, \boldsymbol{\Theta}, \mathbf{y}_a)\\ &= \mathcal{N}(\mu_a(\mathbf{x}, \boldsymbol{\theta}; \mathbf{X}, \boldsymbol{\Theta}, \mathbf{y}_a), s_a^2(\mathbf{x}, \boldsymbol{\theta}; \mathbf{X}, \boldsymbol{\Theta}, \mathbf{y}_a))\end{aligned}$$

where $\mathbf{X}, \boldsymbol{\Theta}$ and $\mathbf{y}_a$ are the training data set corresponding to $\mathbf{x}, \boldsymbol{\theta}$ and y_a, respectively. The predictive mean μ_a and variance s_a^2 are given as follows:

$$\begin{aligned}\mu_a(\mathbf{z}; \mathbf{Z}, \mathbf{y}_a) &= \mathbf{k}_a^{\mathrm{T}}(\mathbf{K}_a + \sigma_a^2\mathbf{I})^{-1}\mathbf{y}_a,\\ s_a^2(\mathbf{z}; \mathbf{Z}, \mathbf{y}_a) &= k_a(\mathbf{z}, \mathbf{z}) - \mathbf{k}_a^{\mathrm{T}}(\mathbf{K}_a + \sigma_a^2\mathbf{I})^{-1}\mathbf{k}_a.\end{aligned}$$

The vector $\mathbf{k}_a$ is denoted as $\mathbf{k}_a = [k_a(\mathbf{z}^{(1)}, \mathbf{z}), \ldots, k_a(\mathbf{z}^{(N)}, \mathbf{z})]^{\mathrm{T}}$. The matrix $\mathbf{K}_a$ is the kernel matrix with $K_{a,ij} = k_a(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ as (i, j) entry. In this paper, the kernel function defined for the calculation of μ_a and s_a^2 is assumed to be the following squared exponential kernel function:

$$k_a(\mathbf{z}, \mathbf{z}') = \alpha_a^2 \exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{z}')^{\mathrm{T}}\left(\mathbf{H}_a^z\right)^{-1}(\mathbf{z} - \mathbf{z})\right)$$

where α_a^2 is the variance of g_a. This selection of the kernel function allows us to obtain the Gaussian predictive distribution with approximation in sense of the 1st and 2nd order moments. Here, $\mathbf{x}$ and $\boldsymbol{\theta}$ are assumed to be independent, accordingly $\mathbf{H}_a^z$ is defined as a block diagonal matrix $\mathbf{H}_a^z = \mathrm{diag}\{\mathbf{H}_a^x, \mathbf{H}_a^\theta\}$, and $\mathbf{H}_a^x$ and $\mathbf{H}_a^\theta$ are diagonal matrices with positive elements which adjust the scale of each dimension of $\mathbf{x}$ and $\boldsymbol{\theta}$, respectively. Note that this model is classified to Multifactor Gaussian Process Models [33]. Hyperparameters to be learned is $\gamma_a = \left\{\alpha_a^2, \sigma_a^2, \mathbf{H}_a^x, \mathbf{H}_a^\theta\right\}$.

5.2.2 Parameter extraction by manifold learning

Here, we consider the following structure. The unknown factor $\boldsymbol{\theta}$ is same for N_ℓ-sample in the training data: only one vector $\boldsymbol{\theta}^{(\ell)}$ is used to represent the dataset $\{\mathbf{y}^{(j)}, \mathbf{x}^{(j)}\}$ for the subset of index j. Here, we consider separated training data set $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_L\}$, $\mathcal{D}_\ell = \left\{\mathbf{x}^{(\ell,i)},\ \mathbf{y}^{(\ell,i)}\right\}_{i=1}^{N_\ell}$, where L is the number of vectors to be learned. We consider to extract the unknown vector set $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)},\ \ldots,\ \boldsymbol{\theta}^{(L)}]$ using the GP model (5.2) with a similar way to the GPLVMs. For our model (5.2), the log-likelihood function is defined as,

$$\log p(\mathbf{y}_a|\mathbf{X}, \boldsymbol{\Theta}, \gamma_a) = -\frac{1}{2}\log\det(\mathbf{K}_a + \sigma_a^2\mathbf{I}) - \frac{1}{2}\mathbf{y}_a^{\mathrm{T}}(\mathbf{K}_a + \sigma_a^2\mathbf{I})^{-1}\mathbf{y}_a - \frac{1}{2}\log(2\pi), \tag{5.3}$$

where $\mathbf{z}^{(i)}$ corresponds to the i-th column of the matrix $\mathbf{Z} \in \mathbb{R}^{d_\mathbf{z} \times N}$ defined as,

$$\mathbf{Z} = \begin{bmatrix} \boldsymbol{\theta}^{(1)} & \cdots & \boldsymbol{\theta}^{(1)} & \cdots & \boldsymbol{\theta}^{(L)} & \cdots & \boldsymbol{\theta}^{(L)} \\ \mathbf{x}^{(1,1)} & \cdots & \mathbf{x}^{(1,N_1)} & \cdots & \mathbf{x}^{(L,1)} & \cdots & \mathbf{x}^{(L,N_L)} \end{bmatrix}, \tag{5.4}$$

where $N = \sum_{\ell=1}^{L} N_\ell$. $\mathbf{X}$ is a $d_\mathbf{x} \times N$ sub-matrix of $\mathbf{Z}$, which is a part of $\mathbf{x}$ in $\mathbf{Z}$, and $\mathbf{y}_a \in \mathbb{R}^N$ is defined as,

$$\mathbf{y}_a = \begin{bmatrix} y_a^{(1,1)} & \cdots & y_a^{(1,N_1)} & \cdots & y_a^{(L,1)} & \cdots & y_a^{(L,N_L)} \end{bmatrix}^{\mathrm{T}},$$

and $\gamma_a = \left\{\mathbf{H}_a^x, \mathbf{H}_a^\theta, \alpha_a^2, \sigma_a^2\right\}$ is a hyperparameter set. Note that $\boldsymbol{\theta}$ is the only unknown variable in the latent variable $\mathbf{z}$, and $\boldsymbol{\theta}^{(\ell)}$ is N_ℓ times included in $\mathbf{Z}$. With holding this structure, $\boldsymbol{\Theta}$ and $\gamma = \{\gamma_1, \ldots, \gamma_{d_y}\}$ are optimized simultaneously by maximizing the sum of the log-likelihood function for all dimensions of $\mathbf{y}$ as

$$(\boldsymbol{\Theta}^*, \gamma^*) \leftarrow \underset{\boldsymbol{\Theta},\gamma}{\arg\max} \sum_{a=1}^{d_y} \log\, p(\mathbf{y}_a|\mathbf{X}, \boldsymbol{\Theta}, \gamma_a). \tag{5.5}$$

5.2.3 Relationship with other methods

Let us discuss the relationship with other methods, Saal et al. [1] and GPLVM [2], by comparing those problem settings. The differences of the problem settings are summarized in Table 5.1. From the point of view of model learning, the true-object parameter $\boldsymbol{\theta}$ is known as training data in Saal et al [1], however, its equivalent parameter $\boldsymbol{\theta}$ is unknown in our problem setting. Next, our model (5.1) seems to be the same model as considered in GPLVM with defining one latent variable $\mathbf{z}$, however, $\mathbf{x}$ which is a part of latent variable $\mathbf{z}$ is known and included in training data. In addition, we have N training data, however, the number of the recognition target objects is L, and generally $L \ll N$. Accordingly, the structure of latent variables corresponding to the observation variable should be fixed as shown in Eq. (5.4).

5.3. Application to object recognition

The proposed method is applicable for the modeling of the observation model in tactile object recognition. In this section, it is described how to apply the proposed method to the tactile object recognition problem.

5.3.1 Background

Object recognition using a robot hand based on tactile information such as pressure, vibration and temperature is a crucial problem (Fig. 5.2). To recognize an object by such a robot, the following procedures are executed: (1) an action to be applied to the object is designed, (2) the robot performs the action to the object, (3) the recognition task is accomplished with the obtained tactile information.

For the efficient recognition, the most important procedure may be (1): we need to plan and execute clever actions (referred to as *exploratory actions*) sequentially so that the resulting sensor data become sufficiently informative (we refer to the action which carries the informative sensor data as an *informative action*). In contrast with the poorly designed (e.g., random or hand-tuned) actions, if we could collect sample data by sequentially performing informative actions, i.e. *active learning*, the required time to accomplish the task would be

Table 5.1. Differences of the problem settings among Saal et al. [1], ours, and GPLVM [2]

	Saal et al. [1] (GPs)	ours	GPLVMs [2]
model	multifactor, $\mathbf{y} = \mathbf{g}(\mathbf{x}, \boldsymbol{\theta})$	multifactor, $\mathbf{y} = \mathbf{g}(\mathbf{x}, \boldsymbol{\theta})$	singlefactor, $\mathbf{y} = \mathbf{g}(\mathbf{z})$
training data set	$\{\mathbf{y}^{(i)}, \mathbf{x}^{(i)}, \boldsymbol{\theta}^{(i)}\}$	$\{\mathbf{y}^{(i)}, \mathbf{x}^{(i)}\}$	$\{\mathbf{y}^{(i)}\}$
latent variables	—	$\boldsymbol{\theta}$	$\mathbf{z}$
number of latent variables	—	same as target object	same as training data

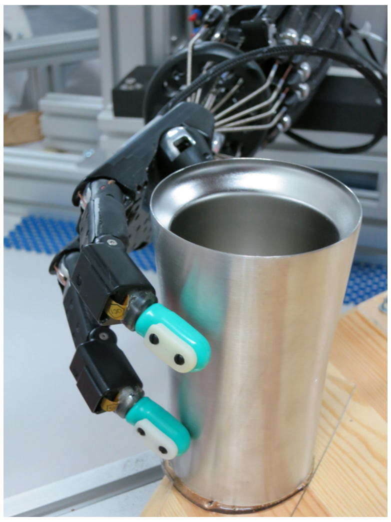


Figure 5.2. Tactile object recognition by a robot hand

drastically reduced. The effectiveness of active learning has been investigated in (e.g. [1, 35, 36, 37, 38, 39, 40]).

5.3.2 Application of the proposed method to object recognition

Active learning requires the *observation model* that relates the observed data to the action and the object to seek informative actions. We consider to learn such a model from training data using Gaussian Processes (GPs) [41], that relates the observed tactile sensor data to the continuous object and action parameters, to enjoy the compatibility of GPs with the active learning based on mutual information, as well as in [1]. However, in the object recognition task, the suitable representations of the objects for the object parameters in the model are not given a priori and they might be strongly task-dependent, unlike in [1]. Besides, using unsuitable object parameters may deteriorate the task performance.

The sensor model is a two factor model: the observation is represented with the exploratory action and the object. If the different exploratory actions are applied to the same object, the observation will be different, and if the same exploratory action is applied to the different objects, the observation will also be different. Thus, the two factor model is suitable to represent the phenomenon.

To train the Gaussian process models, we can obtain the observation and the applied exploratory action as the training data. However, the parameters which represent each object is unknown a priori. This is the situation that one factor in a two factor model is known, and the object manifold learning will be a key to learn this model.

5.3.3 Exploratory action planning

After the object parameters are allocated and the observation model is learned, the informative exploratory action can be designed using the observation model. In addition, let us consider the tactile recognition task using an anthropomorphic robotic hand. Regarding the exploratory action design, to avoid such undesirable situations that the robot might break the object being touched or might get a damage, the compliance of the robot behaviors is important as well as the informativeness of the resulting sensor data.

Thus, we propose to design the exploratory actions using the formulation of the optimal control problem with the robot dynamics. The optimal control can find a control law that minimizes the resulting cost function. We propose the cost function that is composed of two terms: the informativeness and the energy consumption that can promote resulting actions to be compliant. As a criterion of the informativeness, we adopt the mutual information which can be measured by the model obtained by the object manifold learning.

5.3.4 Related works

The object recognition problem is often treated using visual information ([42, 43, 44, 45]). However, the recognition based on visual information might be unrobust because of effects such as occlusion, or the lightning condition in the real environment. Also, the auditory information utilized in ([46, 47]) could be effected in noisy environment. In contrast, the recognition based on tactile information is more robust for such effects as also mentioned in ([48]).

Regarding the allocation of the object parameters, the differential geometry based feature [49] and the force-distance profiles based feature [50] for tactile sensor data could be used as the object parameters. Since these features are strongly related to the physical quantities such as a surface shape and hardness of the object, the suitable features need to be selected for the task a priori. Data-driven feature extraction methods based on dimensionality reduction methods such as Principal Component Analysis [51], Self-Organizing Maps [52, 53], and Maximum Covariance Analysis [54] have also been explored; however, these methods are limited in a single action and unclear how to use for computing informative actions with the notable exception of [38].

Also, regarding the planning of the exploratory actions, most previous studies cannot consider the informativeness and compliance simultaneously since they treat the planning problem of exploratory actions separately from the robot control problem (e.g. [40, 39, 1]). In some previous studies, however, similar methods have attempted to solve different problems as an optimal control problem. The active sensing problem (e.g. [55, 56]), such as field modeling of the environment is addressed with a mutual information criterion. However, contactless sensor, such as a laser rangefinder, or a vision sensor are targeted in those studies, in other words, compliance is not considered in the exploratory action design.

5.3.5 Sequential active learning for object recognition

We treat the object recognition problem as a parameter estimation problem [1]. We assume that each object has the intrinsic parameter called *object parameter*, and this parameter will be sequentially estimated using the tactile sensor data obtained by the exploratory action.

Generally speaking, the procedure of the active object recognition is summarized as follows:

Step 1: Set an initial guess of the object parameter for the (unknown) target object using a probability distribution (called *object's belief*).

Step 2: Design the optimal exploratory action based on the current object's belief.

Step 3: Obtain the observation (tactile sensor data) by executing the designed action to the target object.

Step 4: Update the object's belief based on the observation.

Step 5: Repeat from Step 2 until the variance of the belief becomes sufficiently small.

Step 6: Determine the result as the object which has the nearest object parameter in the database.

5.3.6 Problem formulation

Let us assume the object list to be recognized $\mathcal{O} = \{O_1, \ldots, O_L\}$ is given. We also assume that the robot and tactile sensor for the object recognition are represented as the following the state transition and observation equations:

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\epsilon}, \tag{5.1}$$

where $\mathbf{y} \in \mathbb{R}^{d_y}$ is the d_y-dimensional observation from the robot's sensor, $\mathbf{x} \in \mathbb{R}^{d_x}$ is the d_x-dimensional *action parameter* which parametrizes the exploratory action, $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ is the d_θ-dimensional object parameter, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon), \boldsymbol{\Sigma}_\epsilon = \mathrm{diag}\,\{\sigma_1^2, \sigma_2^2, \ldots, \sigma_{d_y}^2\}$ is the d_y-dimensional Gaussian observation noise.

The problems we need to consider to solve the object recognition problem are listed as follows:

Problem 1 Suppose that the object list $\mathcal{O} = \{O_1, \ldots, O_L\}$ is given. The problems are

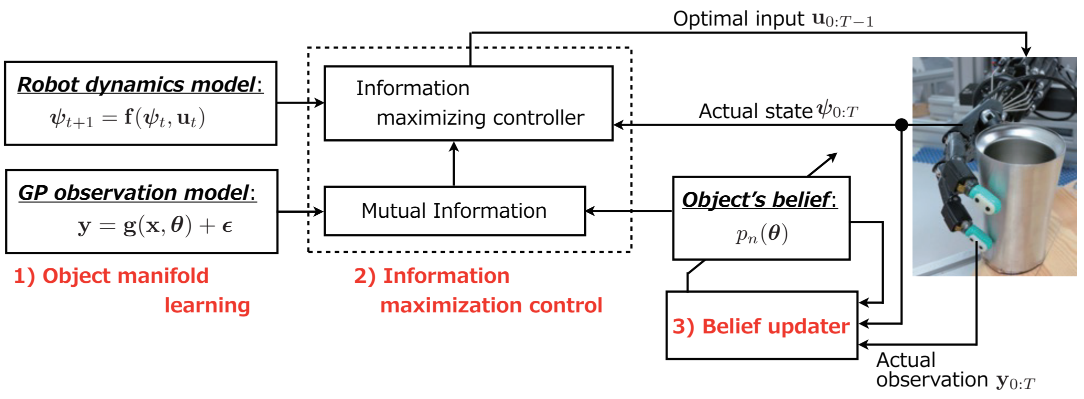
(a) to find the suitable object parameter for each object $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L]$, and

(b) to obtain the observation model (5.1).

Problem 2 Suppose that the state transition model

$$\boldsymbol{\psi}_{t+1} = \mathbf{f}(\boldsymbol{\psi}_t, \mathbf{u}_t), \tag{5.6}$$

where $\boldsymbol{\psi} \in \mathbb{R}^{d_\psi}$ is the d_ψ-dimensional (observable) state of the robot, and $\mathbf{u} \in \mathbb{R}^{d_u}$ is the d_u-dimensional input to the robot with input limits $\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}$ are given. Also, the observation model (5.1) are given. The problem is to find the informative and compliant action sequence.



Optimal input $\mathbf{u}_{0:T-1}$
Robot dynamics model:
$\psi_{t+1} = \mathbf{f}(\psi_t, \mathbf{u}_t)$
Information maximizing controller
Actual state $\psi_{0:T}$
GP observation model:
$\mathbf{y} = \mathbf{g}(\mathbf{x}, \boldsymbol{\theta}) + \epsilon$
Mutual Information
Object's belief:
$p_n(\boldsymbol{\theta})$
1) Object manifold learning
2) Information maximization control
3) Belief updater
Actual observation $\mathbf{y}_{0:T}$

Figure 5.3. Overview of the proposed method. 1) Object manifold learning to obtain a GP sensor model to calculate informativeness of the action, 2) Information maximization with a controller that designs informative and compliant actions, and 3) Object's belief update from obtained observation.

For **Problem 1**, the object manifold learning can be applied. In the following section (Section 5.3.7), we describe how to apply the object manifold learning to the object recognition. Then, the solution for the **Problem 2** is shown in Section 5.3.8. The relationship of the methods is shown in Fig. 5.3.

5.3.7 Object manifold learning with action features

This model can be used in the following 2 cases:

As a Sensor-Action Model

Assuming the desired state sequence $\boldsymbol{\Psi}^d = \{\psi^d_t\}_{t=0}^{T-1}$ is parametrized by $\mathbf{x}$, that is $\boldsymbol{\Psi}^d = \boldsymbol{\Psi}^d(\mathbf{x})$, this model can be regarded as a *Sensor-Action Model* with a tactile feature $\mathbf{y}$ which represents the compressed information of the tactile sequence, $\mathbf{y} \leftarrow \{\mathbf{y}_t\}_{t=0}^{T-1}$.

As a Sensor-State Model

Assuming the robot's state as the instantaneous action, that is $\mathbf{x} = \psi_t$ and $\mathbf{y} = \mathbf{y}_t$, this model can be regarded as a *Sensor-State Model.*

Even the meaning of $\mathbf{x}$ differs, the fundamental structure is same. Also, the difference will not effect to the solution of **Problem 1** with a priori defined $\mathbf{x}$.

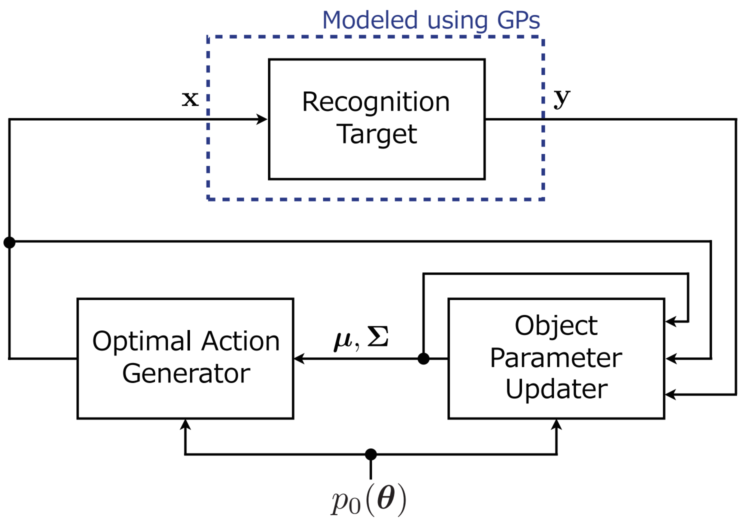


Figure 5.4. Overview of our active object recognition system. We first set the probability distribution of the object parameter $p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ as initial object's belief. Based on the belief, the action $\mathbf{x}$ is computed and performed for the recognition target and accordingly the observation $\mathbf{y}$ is obtained. Object's belief $p(\boldsymbol{\theta})$ is updated using $\mathbf{x}$ and the obtained $\mathbf{y}$.

Thus, we will not consider this difference in the following discussion.

Let us assume the training dataset $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_L\}$, $\mathcal{D}_\ell = \left\{\mathbf{x}^{(\ell,i)}, \mathbf{y}^{(\ell,i)}\right\}_{i=1}^{N_\ell}$ is available. $\mathcal{D}_\ell$ is obtained by executing the exploratory action to the ℓ-th object O_ℓ. Then, using the dataset, we can directly obtain the suitable object parameters by applying the object manifold learning.

5.3.8 Exploratory action generation for active tactile object recognition and belief update

Using the GP model by applying object manifold learning, we construct the active tactile object recognition method, by following [1]. The overview of the whole process is shown in Fig. 5.4.

We use the mutual information [57] as the criterion of informativeness for the exploratory action. The informativeness of exploratory action for each update depends on the current object's belief represented as the probability distribution of an object parameter $\boldsymbol{\theta}$. The mutual information $\mathrm{I}[\boldsymbol{\theta}, \mathbf{y}|\mathbf{x}]$ evaluates the reduced amount of the object parameter's uncertainty when the observation $\mathbf{y}$ is obtained at the state $\mathbf{x}$. In other words, it represents the amount of obtained information.

We utilize the mutual information as the definition of the infomativeness of the action. The mutual information is defined using Kullback-Leibrer Divergence as follows [58],

$$\begin{aligned}\mathrm{I}\big[\boldsymbol{\theta}, \mathbf{y}|\mathbf{x}\big] &\triangleq \mathrm{KL}(p(\boldsymbol{\theta}, \mathbf{y}|\mathbf{x}) \| p(\boldsymbol{\theta}) p(\mathbf{y}|\mathbf{x})) \\ &= \iint p(\boldsymbol{\theta}, \mathbf{y}|\mathbf{x}) \log \frac{p(\boldsymbol{\theta}, \mathbf{y}|\mathbf{x})}{p(\boldsymbol{\theta}) p(\mathbf{y}|\mathbf{x})} \mathrm{d}\mathbf{y} \mathrm{d}\boldsymbol{\theta},\end{aligned} \tag{5.7}$$

and it is also represented using the entropy $\mathrm{H}[\cdot]$ as follows:

$$\mathrm{I}\big[\boldsymbol{\theta}, \mathbf{y}|\mathbf{x}\big] = \mathrm{H}[\boldsymbol{\theta}] - \mathrm{H}[\boldsymbol{\theta}|\mathbf{y}, \mathbf{x}].$$

We can obtain the effective observation for the parameter estimation by controlling the system to the state sequence that maximizes this quantity.

The active tactile object recognition will be achieved as follows: First, the object's belief is initialized as a probability distribution $p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. Next, the exploratory action defined by the action parameter $\mathbf{x}$ is determined by maximizing mutual information, and it is executed for the target object. Object's belief $p(\boldsymbol{\theta})$ is sequentially updated using the observation $\mathbf{y}$ and the action parameter $\mathbf{x}$. Based on updated $p(\boldsymbol{\theta})$, the next exploratory action $\mathbf{x}$ is determined. This procedure is repeated until $n_{\max}$ times updated or terminated if the update of mean is sufficiently small, and then the recognition task is finally achieved by the nearest-neighbor object on the extracted object manifold.

With Sensor-Action Models

If the compliance of the action is guaranteed by the hardware or the pre-designed controller, our interest is only the informativeness.

We consider the probability distribution as object's belief at the n-th update, $p_n(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, for $n = 0, 1, \ldots, n_{\max}$. We first set the initial belief $p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.

The joint distribution between $\boldsymbol{\theta}$ and $\mathbf{y}$ given $\mathbf{x}$ is also given by a Gaussian distribution as follows [59]:

$$p(\boldsymbol{\theta},\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\theta}\\ \mathbf{y}\end{bmatrix}\middle\|\begin{bmatrix}\boldsymbol{\mu}\\ \mathbf{m}(\mathbf{x})\end{bmatrix},\tilde{\boldsymbol{\Sigma}}(\mathbf{x})\right),$$
$$\tilde{\boldsymbol{\Sigma}}(\mathbf{x}) = \begin{bmatrix}\boldsymbol{\Sigma} & \mathbf{C}(\mathbf{x})\\ \mathbf{C}^{\mathrm{T}}(\mathbf{x}) & \boldsymbol{\Phi}(\mathbf{x},\mathbf{x})\end{bmatrix}$$

Enjoying this result, the double integral in Eq. (5.7) can be evaluated analytically, and it is represented using the training data and the hyperparameter as follows:

$$\mathrm{I}\big[\boldsymbol{\theta},\mathbf{y}|\mathbf{x}\big] = -\frac{1}{2}\log\left(\frac{\det\tilde{\boldsymbol{\Sigma}}(\mathbf{x})}{\det\boldsymbol{\Phi}(\mathbf{x},\mathbf{x})\det\boldsymbol{\Sigma}}\right).$$

See Appendix D for the definition of the vector and matrices. Using $p_{n-1}(\boldsymbol{\theta})$, optimal exploratory action parameter at n-th update, $\mathbf{x}_n$, is determined by maximizing the mutual information between $\boldsymbol{\theta}$ and $\mathbf{y}$ defined by $\boldsymbol{\mu}_{n-1}$, $\boldsymbol{\Sigma}_{n-1}$, and $\mathbf{x}$. Enjoying the compatibility of GPs, the mutual information can be evaluated analytically, and the optimal action is obtained by solving the following optimization problem,

$$\mathbf{x}_n = \arg\max_{\mathbf{x}} \mathrm{I}\big[\boldsymbol{\theta},\mathbf{y}|\mathbf{x}\big]$$

When the observation $\mathbf{y}_n$ is obtained, the updated belief $p_{n+1}(\boldsymbol{\theta})$ is obtained by,

$$\boldsymbol{\mu}_{n+1} = \boldsymbol{\mu}_n + \mathbf{C}(\mathbf{x}_n)\boldsymbol{\Phi}(\mathbf{x}_n,\mathbf{x}_n)^{-1}(\mathbf{y}_n - \mathbf{m}(\mathbf{x}_n)),$$
$$\boldsymbol{\Sigma}_{n+1} = \boldsymbol{\Sigma}_n - \mathbf{C}(\mathbf{x}_n)\boldsymbol{\Phi}(\mathbf{x}_n,\mathbf{x}_n)^{-1}\mathbf{C}(\mathbf{x}_n)^{\mathrm{T}}$$

This is based on Bayes' rule, and this is a Extended Kalman Filter like update. Although the Monte Carlo sampling-based updating method is shown in [1], we use the analytical Gaussian updating method for its simplicity.

With sensor-state models: information maximization control

To obtain the informative and compliant exploratory action, we formulate the exploratory action design using the finite horizon optimal control framework [60]. Note that the informativeness needs to be maximized as described for exploratory action design, while the optimal control problem is generally formulated as a minimization problem of the cost. The mutual information is converted into a cost to be minimized as shown later.

Approximate optimal control

We consider the optimal control problem:

Find the control law $\mathbf{u}_t = \boldsymbol{\pi}(t, \boldsymbol{\psi}_t)$ which minimizes the cost function for the system (5.6), that is,

$$\begin{aligned} &\underset{\boldsymbol{\pi}}{\text{minimize}} \;\; J_T \\ &\text{s.t.} \;\;\; \boldsymbol{\psi}_{t+1} = \mathbf{f}(\boldsymbol{\psi}_t, \mathbf{u}_t), \;\;\; \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \end{aligned}$$

where

$$J_T = h(\boldsymbol{\psi}_T) + \sum_{t=0}^{T-1} \ell(t, \boldsymbol{\psi}, \mathbf{u})$$

is the accumulated cost function, $h(\boldsymbol{\xi}_T) \geq 0$ is the terminate cost, and $\ell(t, \boldsymbol{\psi}, \mathbf{u}) \geq 0$ represents the running cost. For the exploratory action design, we set the cost function associated by the informativeness and the energy consumption as follows:

$$\ell(t, \boldsymbol{\psi}, \mathbf{u}) = q(\boldsymbol{\psi}_t) + r(\mathbf{u}_t),$$

where the first term $q(\boldsymbol{\xi}_t)$ is related to the informativeness, and the second term $r(\mathbf{u}_t)$ represents the energy consumption.

We utilize the iterative Linear Quadratic Regulator (iLQR, [61]) as a computational efficient and scalable optimal control solver: the linearized system around the initial state sequence $\bar{\boldsymbol{\psi}}_{0:T}$ corresponds to the initial input sequence $\bar{\mathbf{u}}_{0:T-1}$ are constructed, and the local LQR problem is solved for the linearized system. The iLQR also gives a local feedback gains $\mathbf{L}_t$ along $\bar{\mathbf{u}}_{0:T-1}$, therefore, the control law can be given by $\boldsymbol{\pi}(t, \boldsymbol{\psi}_t) = \bar{\mathbf{u}}_t + \mathbf{L}_t(\boldsymbol{\psi}_t - \bar{\boldsymbol{\psi}}_t)$ [61].

Mutual information-based state cost function

Since larger value of the mutual information indicates more informative, the term $q(\boldsymbol{\xi})$ in the running cost is defined using a certain monotonically decreasing function $v(\boldsymbol{\psi}) \geq 0$ as:

$$q(\boldsymbol{\psi}) = v\Big(\mathrm{I}\big[\boldsymbol{\theta}, \mathbf{y}|\boldsymbol{\psi}\big]\Big).$$

Belief update based on state sequence

The optimal action is planned based on the present belief $p_n(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$ as described before, and then the state sequence $\boldsymbol{\psi}^n_{0:T}$ and observation $\mathbf{y}^n_{0:T}$ are obtained by executing the action for the target object. Based on Bayes' rule

$$p_{n+1}(\boldsymbol{\theta}) = \frac{p\big(\mathbf{y}^n_{0:T}|\boldsymbol{\psi}^n_{0:T}, \boldsymbol{\theta}\big)p_n(\boldsymbol{\theta})}{p\big(\mathbf{y}^n_{0:T}|\boldsymbol{\psi}^n_{0:T}\big)}$$

and Gaussian approximation of the marginal distribution $p\big(\mathbf{y}^n_{0:T}|\boldsymbol{\psi}^n_{0:T}\big)$, the mean and the covariance are updated as follows:

$$\begin{aligned}\boldsymbol{\mu}_{n+1} &= \boldsymbol{\mu}_n + \mathcal{C}_n\mathcal{S}_n^{-1}(\mathcal{Y}_n - \mathcal{M}_n),\\ \boldsymbol{\Sigma}_{n+1} &= \boldsymbol{\Sigma}_n - \mathcal{C}_n\mathcal{S}_n^{-1}\mathcal{C}_n^{\mathrm{T}}\end{aligned}$$

where $\mathcal{C}_n \in \mathbb{R}^{d_{\boldsymbol{\theta}}\times(T+1)d_y}$, $\mathcal{S}_n \in \mathbb{R}^{(T+1)d_y\times(T+1)d_y}$, $\mathcal{Y}_n \in \mathbb{R}^{(T+1)d_y}$, and $\mathcal{M}_n \in \mathbb{R}^{(T+1)d_y}$ are defined as follows:

$$\begin{aligned}\mathcal{C}_n &= \begin{bmatrix} \mathbf{C}(\boldsymbol{\xi}^n_0) & \cdots & \mathbf{C}(\boldsymbol{\psi}^n_T) \end{bmatrix},\\ \mathcal{S}_n &= \begin{bmatrix} \boldsymbol{\Phi}(\boldsymbol{\psi}^n_0, \boldsymbol{\psi}^n_0) & \cdots & \boldsymbol{\Phi}(\boldsymbol{\psi}^n_0, \boldsymbol{\psi}^n_T) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\Phi}(\boldsymbol{\psi}^n_T, \boldsymbol{\psi}^n_0) & \cdots & \boldsymbol{\Phi}(\boldsymbol{\psi}^n_T, \boldsymbol{\psi}^n_T) \end{bmatrix},\\ \mathcal{Y}_n &= \begin{bmatrix} (\mathbf{y}^n_0)^{\mathrm{T}} & \cdots & (\mathbf{y}^n_T)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}},\\ \mathcal{M}_n &= \begin{bmatrix} \Big(\mathbf{m}\big(\boldsymbol{\psi}^n_0\big)\Big)^{\mathrm{T}} & \cdots & \Big(\mathbf{m}\big(\boldsymbol{\psi}^n_T\big)\Big)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}\end{aligned}$$

where $\mathcal{C}_n$ and $\mathcal{S}_n$ are the cross-covariance matrix between $\mathbf{y}^n_{0:T}$ and $\boldsymbol{\theta}$, and the covariance matrix of $\mathbf{y}^n_{0:T}$, respectively. This is an extension of the one-sample belief update law described in [1].

The object recognition is achieved by repeating the optimal action planning and belief updating as described previously. The terminate condition is set by threshold e.g. $\|\boldsymbol{\Sigma}_n\|_{\mathrm{F}}^2 < \epsilon$ with the suitable threshold ϵ, or after $n_{\max}$ times repeat. Finally, the recognition result is obtained as the object corresponding to the nearest object parameter $\boldsymbol{\theta}$ in the database.

5.4. Performance validation of object manifold learning

5.4.1 Experiment 1: synthetic data

Setting

We considered the following nonlinear function,

$$
\begin{aligned}
\mathbf{y} &= \mathbf{h}(\theta, x) + \boldsymbol{\epsilon}, \qquad (5.8)\\
\mathbf{h}(\theta, x) &= \begin{bmatrix} \exp\bigl(-(x-\theta)^2\bigr) \\ \exp\bigl(-(x-\theta^2)^2\bigr) \\ \exp\bigl(-(x-\theta^3)^2\bigr) \end{bmatrix},\\
\boldsymbol{\epsilon} &\sim \mathcal{N}(\mathbf{0}, \mathrm{diag}\,\{0.1, 0.1, 0.1\}),
\end{aligned}
$$

where $\mathbf{y} \in \mathbb{R}^3$ is the observation, $\theta \in \mathbb{R}$ is the true-object parameter (which will be the target of the estimation of object parameter $\hat{\theta}$), and $x \in \mathbb{R}$ is the action parameter.

By setting $\theta = -0.5, 0, 0.2, 0.5$, and x as following normal distribution $\mathcal{N}(0, 1)$, we generated $N_\ell = 100$ training samples for each θ using Eq. (5.8). Therefore, the total number of training samples N is 400.

The object parameter set $\boldsymbol{\Theta}$ was extracted and the GP model was simultaneously learned under the following condition: the dimension of the object parameter was set as $d_\theta = 1$ manually, and initial values of $\boldsymbol{\Theta}$ were randomly chosen, and initial hyperparameters γ_a for all a were manually selected and then optimized numerically. In the recognition task, the belief update was executed for $n_{\max} = 10$. The optimal action parameter $\mathbf{x}$ on each update was determined by numerically maximizing the mutual information.

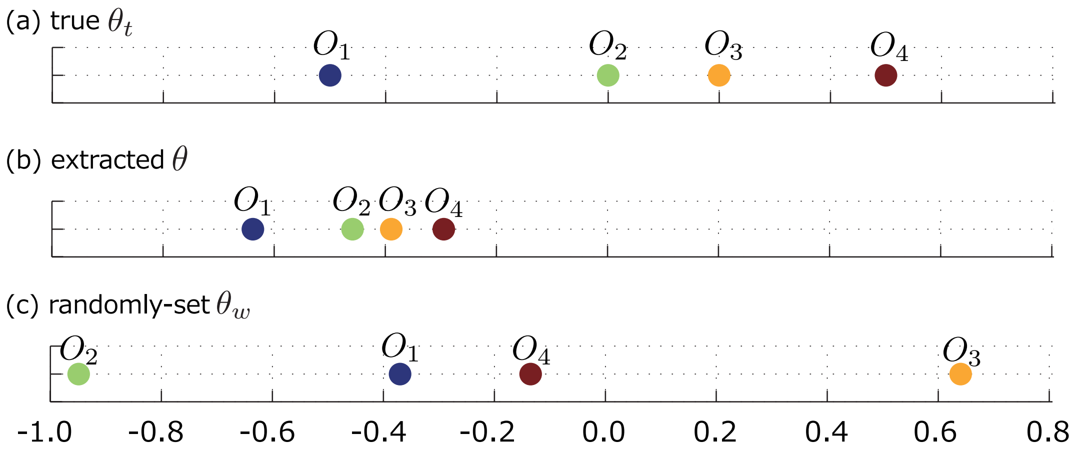


Figure 5.5. Comparison along (a) true-object parameter θ_t, (b) extracted object parameter θ, and (c) randomly-set object parameter θ_w.

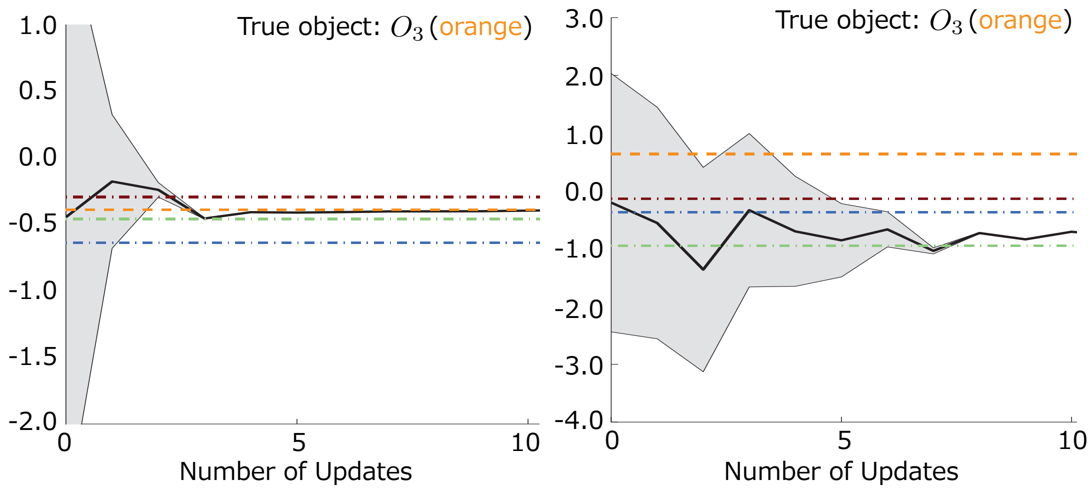
Result of object manifold learning

The extracted object manifold from the training data is shown in Fig. 5.5. The parameter θ_t stands for the true-object parameter, and the parameter θ indicates the extracted object parameters by our method. Since the manifold learning has an ambiguity of the extracted object parameters for its scaling and shifting, we verify the accuracy by the correlation coefficient between θ_t and θ, and the value was 0.9995. Thus, the effectiveness of our object parameter extraction method was confirmed.

Result of Active Object Recognition Simulation

To verify the suitability of the extracted manifold and learned model for active object recognition, we compared its performance of the proposed method for the active object recognition task to that with a GP model using randomly-set object parameter θ_w. The hyperparameters of this model is optimized in the same way as the model with θ. The correlation coefficient between θ_t and θ_w was 0.3449. The mean and covariance of initial belief were set as $\mu_0 = \bar{\theta}$, where $\bar{\theta}$ is the mean of $\boldsymbol{\Theta}$, and $\Sigma = 5$, respectively.

The recognition results are shown in Fig. 5.6. In this recognition simulation, the true object was O_3 represented by the orange dashed line in the figure. As the

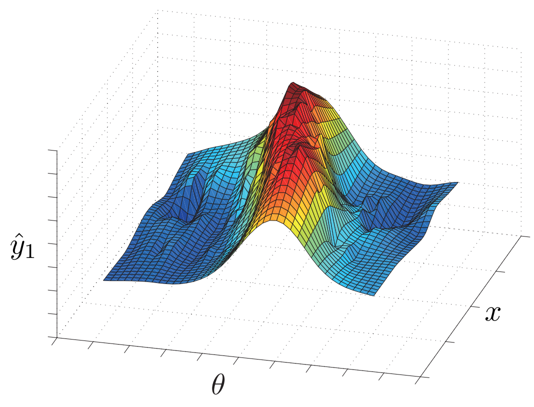


(a) With extracted object parameter $\hat{\theta}$ (b) With randomly-set object parameter θ_w

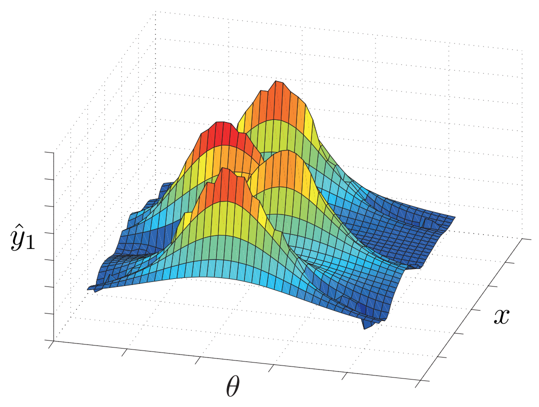
Figure 5.6. Recognition result with the extracted and the randomly-set object parameter θ and θ_w shown in Fig. 5.5. In this simulation, the true object is set to O_3 and its corresponding object parameter is drawn using orange dash-dot line.

result, in the proposed method the estimated object parameter $\hat{\theta}$ successfully and quickly converged to the true value, however, in the comparison the estimated parameter s_w was slower and converged to a wrong value at the end. Let us investigate why this difference occurred by comparing the learned two models. The both models for the 1st dimension of the observation $\hat{y}_1$ are shown in Fig. 5.7. In this figure, the vertical axis $(\hat{y}_1)$ shows the mean of the prediction distribution for the pair (s, x). As you can see, the model with randomly-set object parameters does not have much smoothness in terms of the coordinate x. This would make updating the parameters in active learning difficult since its update law described in Section 3 involves local linearization.

Consequently, the suitability of our proposed object's belief update method and model learning method for active object recognition was verified with the synthetic data.



(a) With extracted object parameter θ



(b) With randomly-set object parameter θ_w

Figure 5.7. Comparison of the GP model of $\hat{y}_1$ which is the 1st dimension of observation.

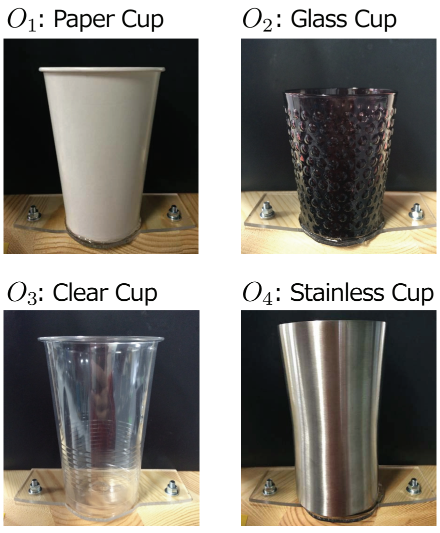
5.4.2 Experiment 2: active tactile object recognition

Setting

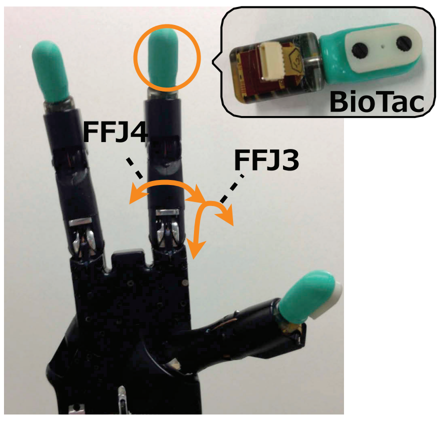
We prepared $L = 4$ objects as recognition targets shown in Fig. 5.8(a). This experiment was done with the robot hand (Shadow Dexterous Hand by Shadow Robot Company), and the tactile sensor mounted on its fingertip (BioTac by SynTouch) shown in Fig. 5.8(b). The whole flow of the experiment is shown in Fig. 5.9. While this robot hand has 12 DoFs, in this experiment we focused on 2 DoF, FFJ3 and FFJ4, as shown in Fig. 5.8(b). These joints can generate actions that correspond to inflective and horizontal movements of the index finger, respectively. This robot hand with the sensor is controlled using Robot Operating System (ROS) [62]. Its control rate and sensor data collection frequency were both 1000 [Hz]. We developed the automated object switching system (1DoF) as shown in Fig. 5.10, that can mount 10 objects at maximum and its angular resolution is 0.2 [deg]. Therefore, the system allowed to automatically collect training data for different objects.

We describe below the details of the action parameter, tactile sensor data, and the setting of model learning and recognition simulation.

Action parameters We defined the action parameter based on Dynamic Movement Primitives (DMPs). DMPs represent the trajectory $\mathcal{Y}$ using nonlinear



(a)



(b)

Figure 5.8. Settings for Experiment 2 of object manifold learning. (a) Recognition target objects. O_1: disposable paper cup. O_2: bumpy glass cup. O_3: disposable clear cup. O_4: stainless cup. (b) 12DoF robot hand and tactile sensor. We use 2 DoF, FFJ3 and FFJ4 corresponding to inflective and horizontal movements of the index finger, respectively. On its fingertips, the BioTac sensor is mounted to obtain the tactile information.

function e.g. $\mathcal{Y} = F(\mathbf{w}, \mathcal{P})$, where $\mathbf{w}$ is the parameter which defines the shape of trajectory, and $\mathcal{P}$ is the hyperparameter set containing time constants, goal state, basis functions, and so on. See [63] for the details.

In this experiment, we first obtained the basic parameter $\mathbf{w}_{\mathrm{teach}}$ from a teaching trajectory, and generated the new trajectory using the new parameter $\mathbf{w} = \mathbf{w}_{\mathrm{teach}} + \mathbf{x}$, where $\mathbf{x}$ is set as the action parameter. The detail is as follows: by using a cyberglove as the master system, we controlled the robot hand as the slave system so that the fingertip pushed into the object, and then started to slide the object subsequently using FFJ3 and FFJ4. The duration of each action is around 14.4 [sec]. Then, the recorded robot trajectories were approximated by two DMPs with 25 basis functions, and each basic parameter $\tilde{\mathbf{w}}^{\mathrm{FFJ3}}_{\mathrm{teach}} \in \mathbb{R}^{25}$ and $\tilde{\mathbf{w}}^{\mathrm{FFJ4}}_{\mathrm{teach}} \in \mathbb{R}^{25}$ were learned using a least square method. To reduce the dimension of the action parameter, we selected three dominant parameters out of 25 parameters, $\mathbf{w}^{\mathrm{FFJ3}}_{\mathrm{teach}} \in \mathbb{R}^{3}$ and $\mathbf{w}^{\mathrm{FFJ4}}_{\mathrm{teach}} \in \mathbb{R}^{3}$, respectively. Finally, we obtained $\mathbf{w}_{\mathrm{teach}} = [(\mathbf{w}^{\mathrm{FFJ3}}_{\mathrm{teach}})^{\mathrm{T}}, (\mathbf{w}^{\mathrm{FFJ4}}_{\mathrm{teach}})^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6}$. Therefore, the dimension of the action parameter turned into $d_x = 6$.

Tactile information BioTac sensor gives pressure, vibration, and temperature as tactile information. In this experiment, $d_y = 5$ dimensional tactile feature was used; vibrations (2 dimensional)[*], pressure, heat flux, and temperature, all of which were obtained by using ROS. Since one action parameter corresponds to one trajectory (time series), we defined $\mathbf{y}$ as the mean of the time series of tactile sensor data and used for the following experiments.

Model learning and recognition simulation We collected $N_\ell = 100$ training data from each target object for constructing a model using the actual robot hand. Using $N = 400$ training data in total, the object parameter set $\mathbf{\Theta}$ were extracted and the GP model was simultaneously learned with $d_\theta = 2$ which was manually selected. In this model learning, a differential evolution scheme [64] was used since the marginal likelihood function has

[*]Note that this tactile sensor has only 1 vibration sensor, however, the measurement frequency of vibration data is 2000 [Hz], while the collection frequency is 1000 [Hz]. The 2nd dimension of the vibration data contains 0.5 [ms] late behind the 1st data.

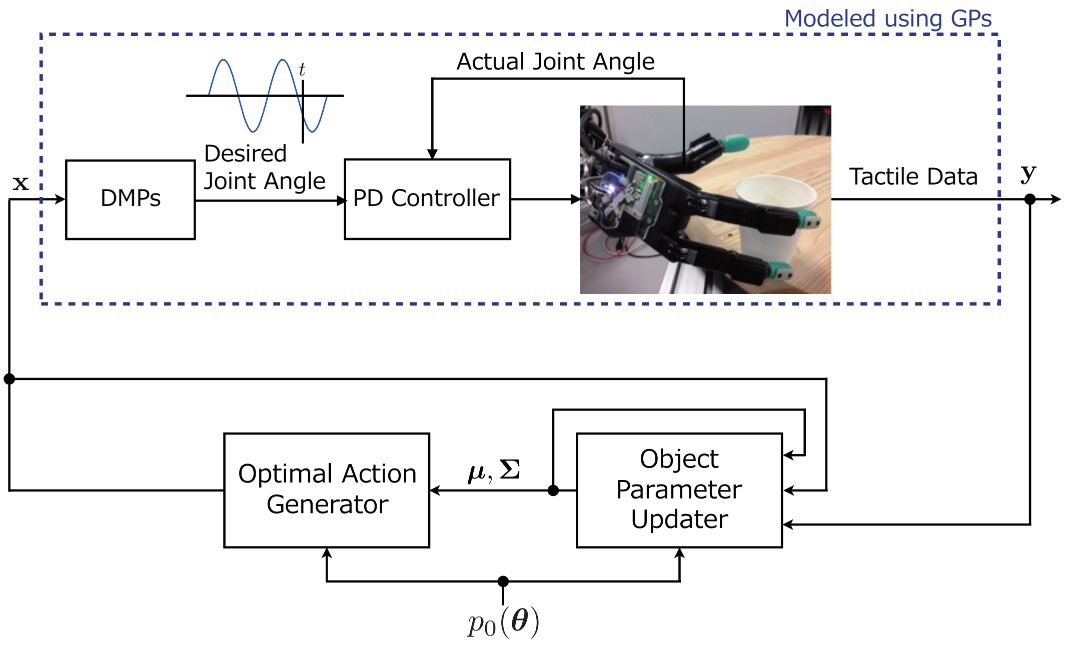
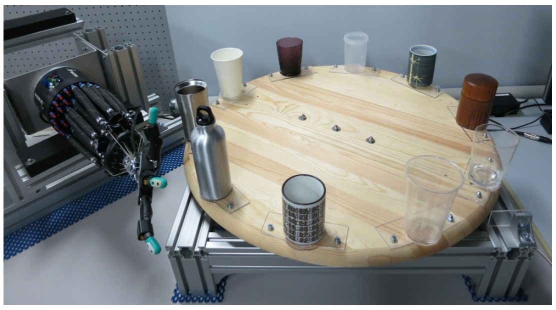


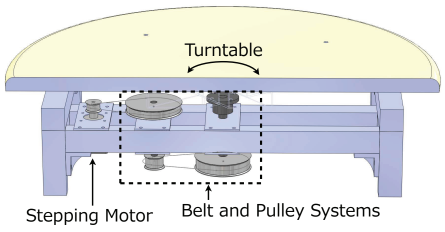
Figure 5.9. Overview of recognition experiment with the real robot hand shown in Fig. 5.8(b). The action parameter $\mathbf{x}$ is converted to a trajectory via DMPs. The robot follows the desired joint angle in the converted trajectory using a PD controller. Consequently, the tactile data $\mathbf{y}$ is obtained. Remaining parts are same as described in Fig. 5.4.

local maxima, and this problem is more serious as compared to Experiment 1 that has less parameters to be optimized. Other settings in model learning were the same as in Experiment 1. To execute the recognition task, the observation $\mathbf{y}$ was sampled from the constructed GP model, and the mean and covariance of initial belief were set to $\boldsymbol{\mu}_0 = \bar{\boldsymbol{\theta}}$ where $\bar{\boldsymbol{\theta}}$ is the average of $\boldsymbol{\Theta}$, and $\boldsymbol{\Sigma} = \mathrm{diag}\,\{5, 5\}$, respectively. The belief update was executed for $n_{\max} = 15$. The optimal action parameter $\mathbf{x}$ on each update is set by numerical maximization.

Result of object manifold learning

Fig. 5.11 shows the extracted object manifold.


(a) Overview



(b) Cross section by its CAD model

Figure 5.10. Object switching system for obtaining training data. The turntable (wooden round table) turns by the stepping motor and it is controlled using the same computer for the robot hand. The torque from the stepping motor is transferred to the turntable through a belt and pulley systems to amplify it.

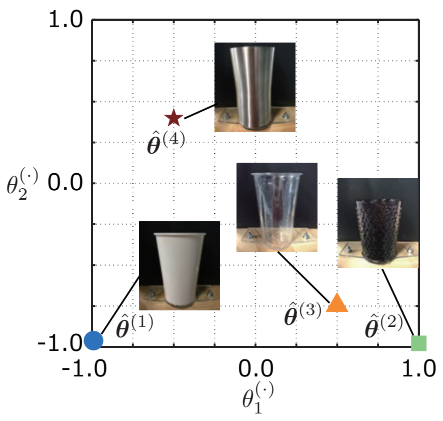
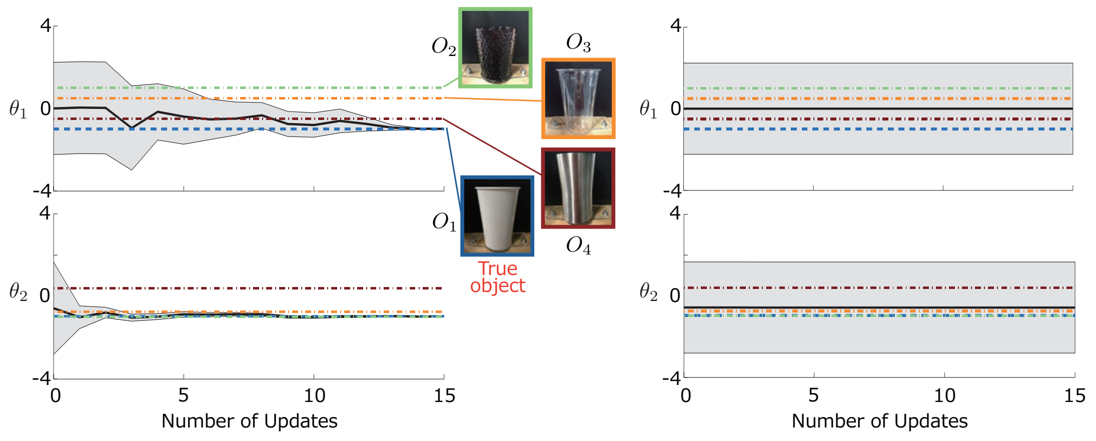


Figure 5.11. Extracted object parameters of 4 objects



(a) with active learning (b) with passive learning

Figure 5.12. Transitions of mean and standard deviation of object's belief at n-th update, $p_n(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$ during learning. True object in this simulation is O_1 (paper cup, represented using blue dash-dot line). The black line represents the mean of $\boldsymbol{\mu}_n$, and the gray area around the black line corresponds to the standard deviations (the square root of diagonal elements of $\boldsymbol{\Sigma}_n$).

Let us discuss the result; O_4 (stainless cup) is located far from other three objects since its heat characteristic is particularly different. O_1 (paper cup) and O_3 are located nearly based on their similar hardness. The appropriateness of these placement will be thoroughly validated with more objects in the near future.

Result of active object recognition simulation

The recognition simulation result and the trajectories corresponding to the computed action parameters at each update are shown in Figs. 5.12 and 5.13, respectively. In Fig. 5.12 the estimation result of both active learning and passive learning are shown and first 5 trajectories are also shown in Fig. 5.13 because of limited space. By using active learning, the recognition is successfully achieved since the mean of $p(\boldsymbol{\theta})$ is converging to the true object O_1 as you can see in Fig. 5.12(a). Comparing the result of active learning, passive learning (Fig. 5.12(b)) does not converge to the true object parameter. We consider this is because of high dimensionality of action space. Fig. 5.13 shows the generated trajectories of exploratory actions on each update. Regarding the result of active learning, in the first update, the movement of FFJ4 is a bit stronger and of FFJ3 is a bit weaker as compared to the teaching trajectory. The difference between the generated movement and the teaching trajectory, however, this movement drastically reduces the variance of $\hat{\theta}_2$ (related to the uncertainty of the estimation). The following movements gradually reduce the variance of $\hat{\theta}_1$. The movements are larger in passive learning as opposed to active learning, however, these movements did not make the variance small. As a result, active learning generates the proper movements to obtain the most informative observation.

From these results, it was confirmed that the model constructed by our proposed method works well for the active object recognition problem even with a real robot data.

5.5. Performance validation of information maximization control

5.5.1 Experiment 1: with physical simulator

Simulation settings

We verify the effectiveness of our proposed scheme using the one link robot arm model shown in Fig. 5.14. The joint range is limited to $-\pi/2 \leq q \leq \pi/2$, and its equation of motion are discretized in a Euler integration manner with the sampling time $\Delta t = 0.01$[s]. We assume that the 2 DoF pressure sensor is mounted on the tip of the arm in order to obtain the observation. The reaction force model f_1 with the spring K and the damper D for the object as shown in Fig. 5.14 is supposed for the horizontal axis, and the dynamic friction model f_2 with the coefficient of dynamic friction μ' is also assumed for the vertical axis as follows:

$$\begin{aligned} f_1 &= -(K\xi_x + D\dot{\xi}_x), \\ f_2 &= -\mathrm{sign}(\dot{\xi}_y)\mu' f_1 \end{aligned}$$

where $\boldsymbol{\xi} = [\xi_x, \xi_y]^{\mathrm{T}}$ is the tip position of the arm.

In this experiment, the object recognition problem is regarded as the damper

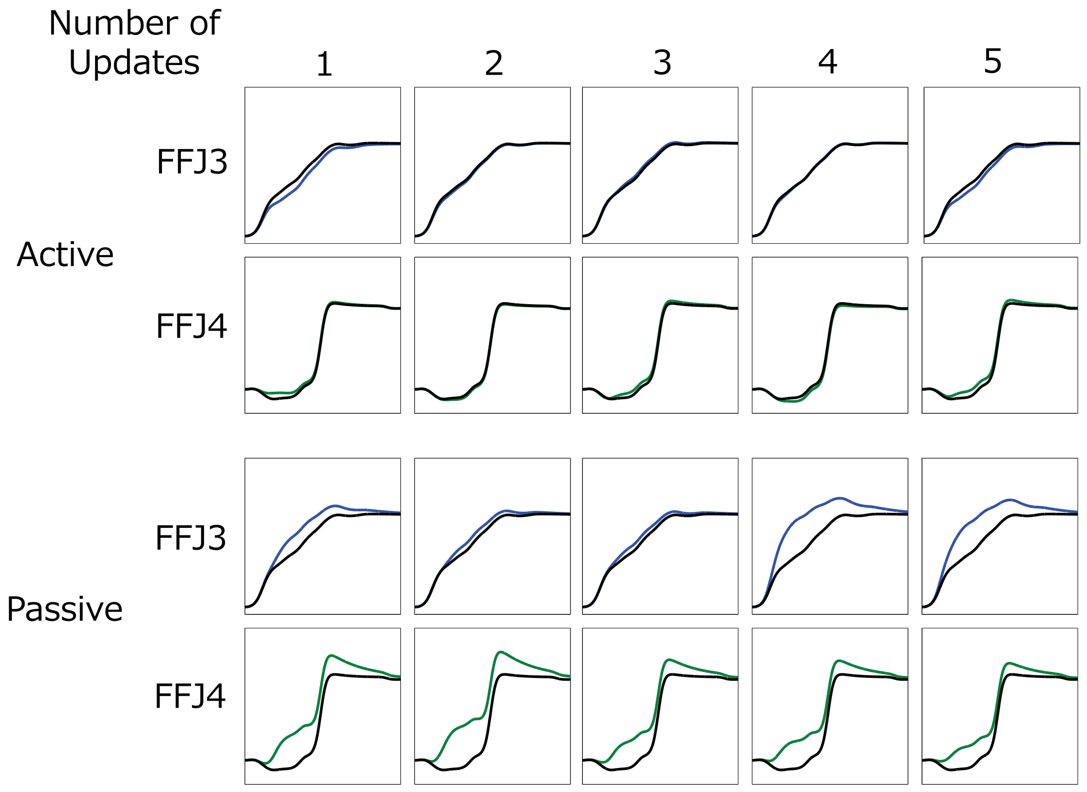


Figure 5.13. Generated trajectories of exploratory actions on each update in recognition. Blue and green lines represent the trajectory generated using $\mathbf{w}_{\text{teach}} + \mathbf{x}_t$, and black lines represent the trajectory generated using $\mathbf{w}_{\text{teach}}$. Time is on the x-axis and the joint angles on the y-axis. The movements in passive learning are larger as opposed to active learning, however, these movements are not informative.

coefficient estimation problem: the object parameter $\theta = D$ is estimated using the exploratory action.

Let us describe how to learn the GP observation model. The spring coefficient and the dynamic friction coefficient are fixed as $K = 1$ and $\mu' = 0.5$, respectively, and we prepare 3 target objects $D \in \{1, 3, 5\}$. The observation $\mathbf{y} = \begin{bmatrix} f_1, f_2 \end{bmatrix}^{\mathrm{T}}$ and the state $\boldsymbol{\psi} = \begin{bmatrix} q, \dot{q} \end{bmatrix}^{\mathrm{T}}$ are defined, and the training data $\mathcal{D}$ is constructed as follows: the range of state is set to $-\pi/2 \leq q \leq \pi/2$ and $-15 \leq \dot{q} \leq 15$, and a 15×15 grid is arranged at equal intervals on the range. We obtain the observation $\mathbf{y}$ for each grid point. The total number of training data is $N = 675$. The object's belief update is executed $n_{\max} = 10$ times.

Next, let us explain the settings for the information maximization control. The initial state is fixed to $\boldsymbol{\psi}_0 = \begin{bmatrix} -\pi/2, 0 \end{bmatrix}^{\mathrm{T}}$, and the length of the exploratory action is set to $T = 100$. The initial input sequence is set to $u_t = 3$ for $t = 0, 1, \ldots, T-1$. The running cost based on the belief $p_n(\theta)$ is defined as

$$\begin{aligned} \ell_n(t, \boldsymbol{\psi}, u) = &\exp\Big(-\rho^{(n)} \mathrm{I}\big[\theta, \mathbf{y} | \boldsymbol{\psi}\big]\Big) + r u^2 \\ &+ 1000 \exp(-10(15 - \dot{q})) \\ &+ 1000 \exp(-10(15 + \dot{q})) \end{aligned}$$

where $\rho^{(n)} > 0$ is a constant in order to change the maximization problem of the mutual information to the minimization problem, and experimentally $\rho^{(n)} = 10 \exp(-0.2 n^2)$ is used since the magnitude of mutual information will decrease by belief's updates. The 3rd and 4th terms in the running cost play a role of the penalty term as the angle velocity does not exceed the range of training data. The terminate cost is set to $h(\boldsymbol{\psi}) = \ell_n(T, \boldsymbol{\psi}, 0)$. The initial object's belief is given $p_0(\theta) = \mathcal{N}(3, 5^2)$, where its mean is equal to the mean of the object parameter in the training data.

Result

Firstly, the mutual information distribution based on the initial object's belief $p_0(\theta)$ computed using the GP observation model is shown in Fig. 5.15. The force from the damper depends on the velocity in the direction of the horizontal axis $\dot{\xi}_x$. The velocity $\dot{\xi}_x$ is gotten zero when the angle $q = 0$ and larger force from

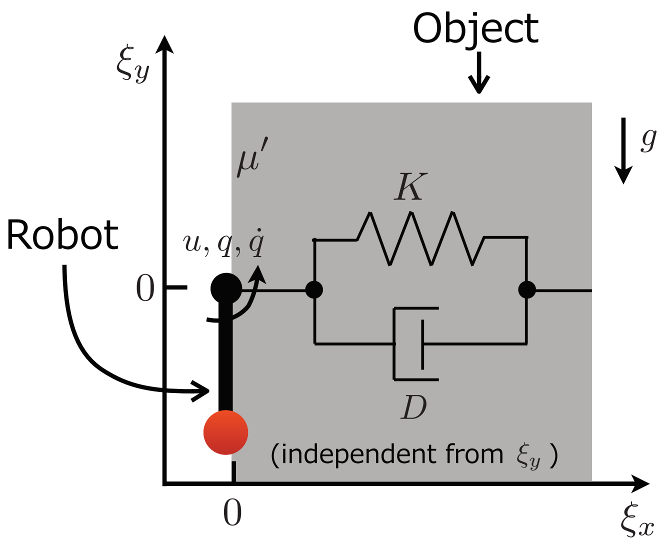


Figure 5.14. Problem setting for Experiment 1 of Information Maximization Control. A tactile sensor is mounted on the tip of 1 DoF robot arm. As the object model, the spring-damper model is assumed for the horizontal direction, and the dynamic friction model is also assumed for the vertical direction.

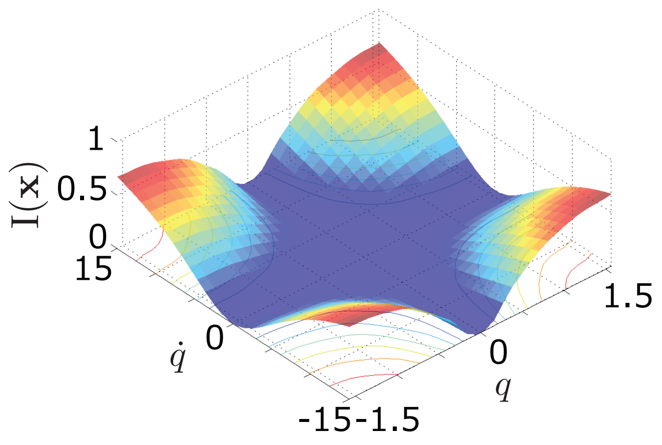


Figure 5.15. Distribution of the mutual information based on the initial object's belief $p_0(\theta) = \mathcal{N}(3, 5^2)$.

the spring is observed. Accordingly, the information about the damper could be buried in other information. Whereas, more information could be obtained when the absolute value of the angle $|q|$ is close to $\pi/2$. Consequently, we regard this distribution as appropriate.

Secondly, the recognition result is shown in Fig. 5.16. The input torque sequences obtained at $n = 0$ by our method are shown in the upper row of Figs. 5.16(a)-(b). For both cases with different values in r, the energy efficient and compliant exploratory actions are generated by the proposed method and the object recognition is successfully achieved. The balance in between the informativeness and energy efficiency is adjusted by the r: the larger value in r (Fig. 5.16(b)) generated energy-efficient actions, however, it is less informative as evidenced with the slower convergence of the belief updates than the smaller value (Fig. 5.16(a)). It was also confirmed that all the elements of local feedback gains $\mathbf{L}$ were relatively small for all the cases. Therefore, the generated controllers are compliant.

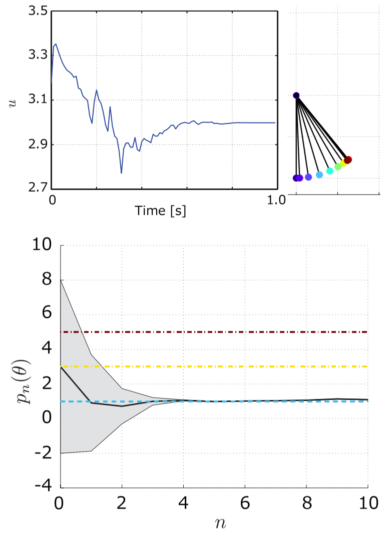
As the comparison, we implemented the PD controllers which generate the control input for achieving the desired state $\boldsymbol{\psi}_d = [\pi/2, 15]^{\mathrm{T}}$, and here this planning has done separately from control problem. The most informative action is obtained using the high gain PD controller as shown in Fig. 5.16(c) but the obtained action is energy inefficient since the large torque sequence is generated. In contrast, a more energy efficient action is obtained using the lower gain PD controller as shown in Fig. 5.16(d); nevertheless the convergence of the belief is slower as compared to the other methods since the planed action is infeasible by the controller.

These experimental results show that our proposed method can generate energy efficient and compliant exploratory behaviors.

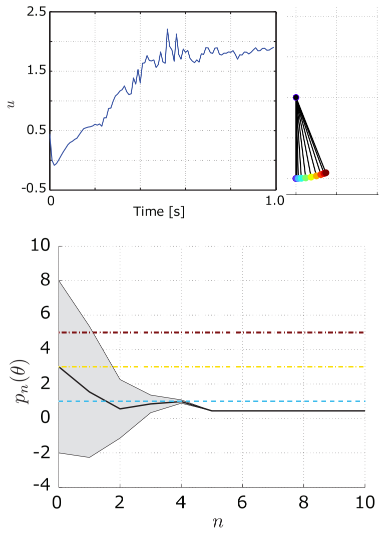
5.5.2 Experiment 2: with actual robot

Experimental settings

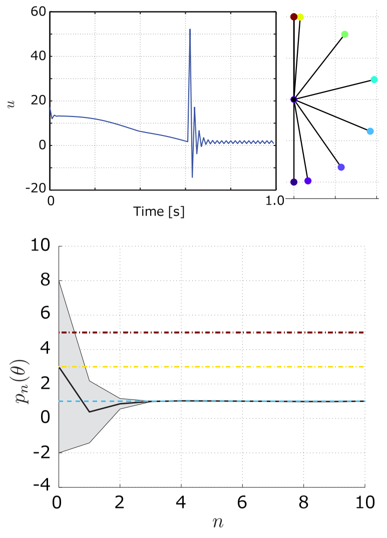
The effectiveness of our proposed scheme is validated by the robot hand system shown in Fig. 5.10 as used in previous section. We prepared $L = 4$ objects as recognition targets shown in Fig. 5.17(a). This experiment was done with the robot hand (Shadow Dexterous Hand by Shadow Robot Company), and the tactile sensor mounted on its fingertip (BioTac by SynTouch) shown in Fig. 5.17(b). While this robot hand has 12 DoF, in this experiment we focused on 2 DoF, FFJ3



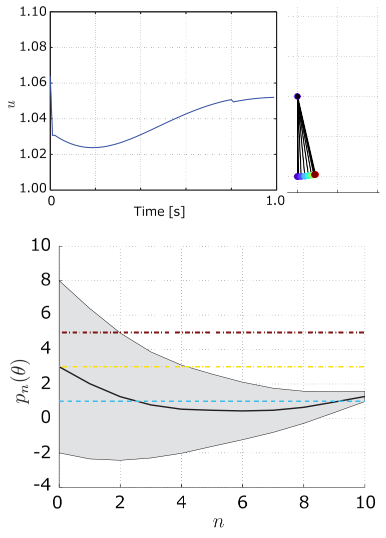
(a) $r = 0.1$



(b) $r = 30$

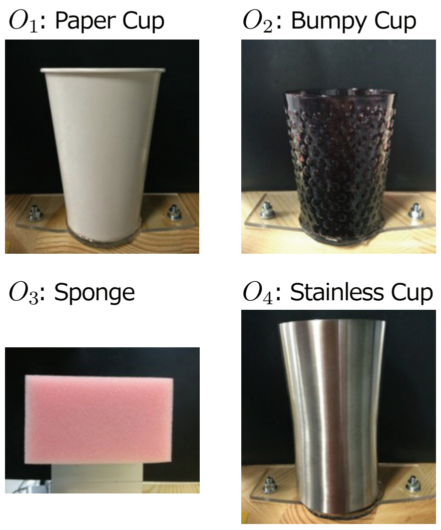


(c) High Gain PD

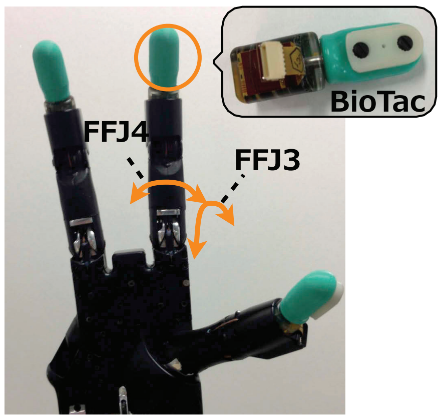


(d) Low Gain PD

Figure 5.16. Results in Experiment 1 of Information Maximization Control. Regarding (a-b), more compliant but less informative actions are designed for the larger value in r. (c) This result indicates that the high gain PID controller gives informative actions. However, these are not compliant. (d) In contrast with (c), the actions are not informative but compliant. [Upper row] The left figure shows the input torque sequence of obtained exploratory action at $n = 0$, and the right figure shows the trajectories of arm for each action corresponds to the torques shown in the left figure. The color of the tip corresponds to the time. [Lower row] The horizontal axis shows the number of updates of the object's belief and the initial belief is set to $p_0(\theta) = \mathcal{N}(3, 5^2)$. The true object parameter is $D = 1$ shown using light blue line. The black line and gray region stand for the mean and standard deviation of the belief $p_n(\theta)$, respectively.



(a) Target objects



(b) Robot hand and tactile sensor

Figure 5.17. Experimental settings for Experiment 2 of Information Maximization Control. (a) Target objects in this experiment. (b) 12 DoF robot hand and tactile sensor. Each joints are driven by pneumatic artificial muscles placed antagonistically. We use 2 DoF corresponding to inflective (pushing) and horizontal (rubbing) movements of the index finger, respectively. The tactile sensor is mounted on the fingertip.

and FFJ4, as shown in Fig. 5.17(b) because of the limitation of the scalability. These joints can generate actions that correspond to inflective and horizontal movements of the index finger, respectively.

The dynamics model (5.6) of this robot hand is difficult to derive analytically because of the complex properties of the pneumatic artificial muscles. Instead, we identified it from training data using a nonlinear discrete-time ARX model whose nonlinearity is wavelet network and one-layer sigmoid network and whose sampling period is set to 0.01s. This are implemented in the MATLAB System Identification Toolbox. The state $\boldsymbol{\psi}$ here is $d_\psi = 4$ dimensional $\boldsymbol{\psi} = \left[q_{\mathrm{FFJ3}}, \dot{q}_{\mathrm{FFJ3}}, q_{\mathrm{FFJ4}}, \dot{q}_{\mathrm{FFJ4}}\right]^{\mathrm{T}}$, where q_{FFJ3} and q_{FFJ4} are the joint angle of FFJ3 and FFJ4, respectively, and $\dot{q}$ stands for each joint velocity. The joint angle ranges of FFJ3 and FFJ4 are $0 \leq q_{\mathrm{FFJ3}} \leq \pi/2$ and $-\pi/9 \leq q_{\mathrm{FFJ4}} \leq \pi/9$, respectively. Here, each input $u_j, j \in \{\mathrm{FFJ3}, \mathrm{FFJ4}\}$ is defined as the difference

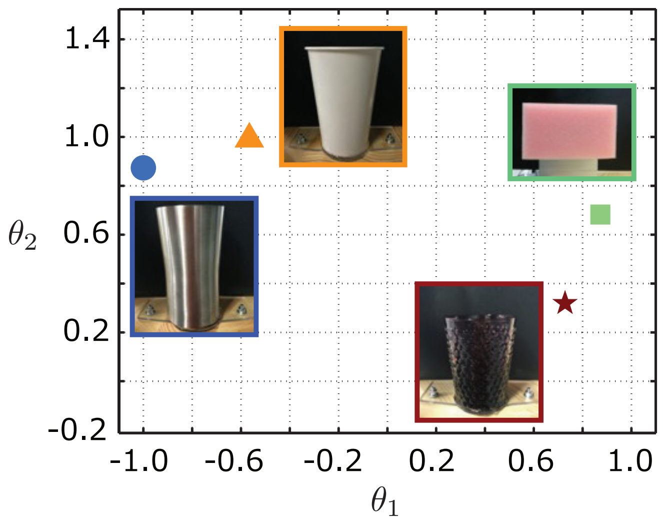


Figure 5.18. Object parameters obtained by object manifold learning. In this experiment, the object parameters for all the objects are identified.

between the desired angle q_j^d and the actual angle q_j, $u_j = q_j^d - q_j$. In the training data collection, the desired angle was set to their maximum and minimum joint angles alternatively. The independency between joints are assumed, therefore, two dynamics models are separately learned for those joints. The total number of training data is 26,765.

The BioTac sensor gives pressure, vibration, and temperature as tactile information. In this experiment, $d_y = 3$ dimensional tactile feature was used; 1-dimensional pressure data and 2-dimensional impedance data, all of which were obtained by using ROS. We collected 100-sample training data for each objects and the number of whole training data was 400. This data collection was done as follows: We design a random trajectories for each joint, and tens of thousands data is collected. And then, 100 data for each object is selected randomly. The object parameters were given by the object manifold learning scheme as shown in Fig. 5.18.

Here, the initial state was fixed to $\boldsymbol{\psi}_0 = \left[\pi/12, 0, 0, 0\right]^{\mathrm{T}}$, and the length of the exploratory action was set to $T = 100$. The initial input sequence is set to

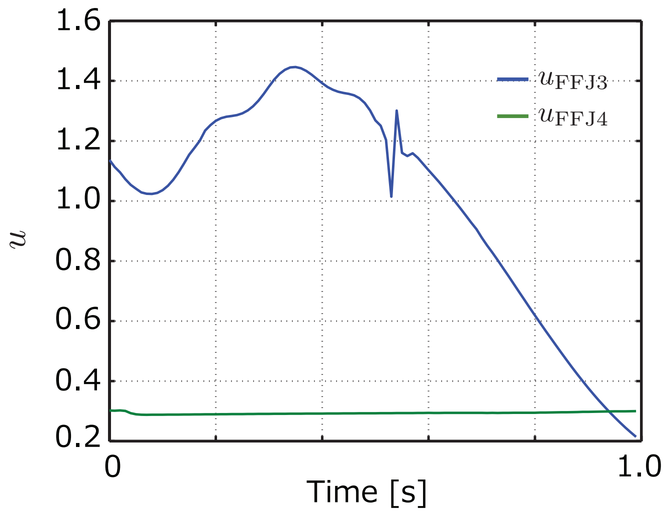


Figure 5.19. Input of obtained action in Experiment 2 of Information Maximization Control. Positive values indicate that the joint moves a direction which increases the joint angle. The joint FFJ3 (blue) generates pushing movements, and the joint FFJ4 (green) generates rubbing movements. This input sequence lets the robot push and rub the object simultaneously.

$u_t = [0.2, 0.3]^{\mathrm{T}}$ for $t = 0, 1, \ldots, T-1$. The running cost was set to

$$\ell(t, \boldsymbol{\psi}, \mathbf{u}) = 10 \exp\Big(-\rho \mathrm{I}\big[\theta, \mathbf{y}|\boldsymbol{\psi}\big]\Big) + \mathbf{u}_t^{\mathrm{T}} \mathbf{R} \mathbf{u}_t$$
$$+ \sum_{j=1}^{4} \Big(\exp(-10(\psi_{j,\max} - \psi_j)) + \exp(-10(-\psi_{j,\min} - \psi_j)) \Big)$$

where $\rho = 30$, $\mathbf{R} = 0.1\mathbf{I}$, $\psi_{j,\max}$ and $\psi_{j,\min}$ stand for the maximum and minimum values of the j-th entry of the state in the training data for GP model construction, respectively.

The object's belief $p_0(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ was given by $\boldsymbol{\mu}_0 = [0.8, 0.5]^{\mathrm{T}}$ and $\boldsymbol{\Sigma}_0 = 0.2^2\mathbf{I}$, which means that it is uncertain whether the target object is O_2 (Dumpy Cup) or O_3 (Sponge). The exploratory action is designed under the conditions.

Results

The computed input sequence $\mathbf{u}$ is shown in Fig. 5.19. Here, the joint moves to a direction which increases its joint angle if the positive values are inputed, because u_j stands for $u_j = q_j^d - q_j$. As you can see in Fig. 5.19, the robot starts to push and rub simultaneously. Intuitively speaking, to discriminate dumpy cup and sponge, the finger should push the object to confirm its stiffness, and also rub the object to check the dumpiness. The obtained action is shown in Fig. 5.20. The upper row of Fig. 5.20 shows the action sequence at $t = 0, 20, \ldots, 100$ and the lower row shows the pose difference from the pose at $t = 0$. Inflective movements were firstly observed ($t = 0$ to $t = 40$) due to hardware properties and the initial state; the joint's movement gets slower if its angle is close to its limit, and the margin between the initial state and the angle limit of FFJ3 is wider than FFJ4. And then horizontal movements are observed ($t = 40$ to $t = 100$). It was also confirmed that the generated controllers are compliant since all the elements of local feedback gains $\mathbf{L}$ were relatively small. This movement can be interpreted as a suitable exploratory action to reduce the uncertainty in between the objects O_2 and O_3. The further investigation is required, but these results suggest that the effectiveness of our proposed method for exploratory action design in real environment.

5.6. Summary

We proposed a partial manifold learning method. It is the manifold learning method for two factor models whose one factor is unknown. The proposed method can extract the unknown factor considering the effect of known factor and observation. The method is applied to the object recognition tasks. Then, we call the proposed method object manifold learning. To solve the recognition task quickly, we need to construct a model for the action optimality evaluation. The object manifold learning obtains the suitable object parameters (corresponds to the unknown factor), and GP sensor model is obtained as the result. Our contributions in the point of view of object recognition are summarized as follows:

(1) a data-driven approach for obtaining the object parameters is proposed, i.e.

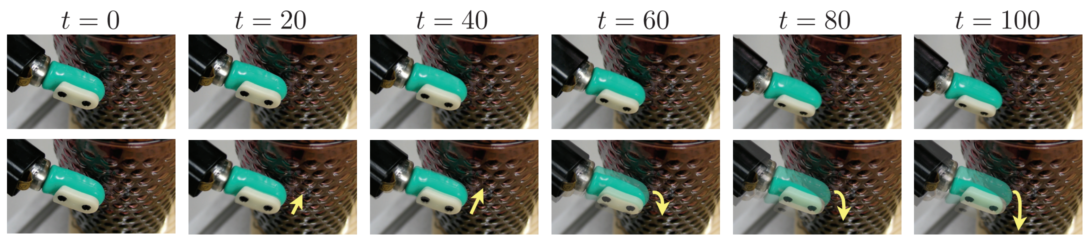


Figure 5.20. Obtained action in Experiment 2 for Information Maximization Control. Intuitively speaking, to discriminate dumpy cup and sponge, the finger should push the object to confirm its stiffness, and also rub the object to check the dumpiness. The upper row shows the action sequence at $t = 0, 20, \ldots, 100$ and the lower row shows the pose difference from the pose at $t = 0$. The robot starts to push and rub simultaneously. Inflective movements are firstly observed ($t = 0$ to $t = 40$) due of hardware properties and the initial state, and then horizontal movements are observed ($t = 40$ to $t = 100$).

object manifold learning with action features.

(2) With the object manifold learning, generalization of the GP based active learning method proposed in [1] is achieved for object recognition problems.

(3) An optimal control formulation is proposed for the exploratory action design considering both the informativeness and the compliance. The method is called information maximization control.

The effectiveness of our proposed method was verified through experiments with synthetic and real robot data.

Let us discuss several directions for future work. We validated our method using 4 actual objects, We are now conducting the experiments with more objects for more concrete validation of our method. In addition, continuous actions are considered in our method in contrast with using discrete actions [35, 36, 39, 40] (e.g. grasp, shake). Comparison between these methods and our method with an action-sensor model would be interesting for future work.

An extension of scalability of the method is also our future work. The high dimensional action space could make the model learning and the optimal action search intractable since a huge number of training data are required. Applying a concept of muscle synergies [65] or other dimensionality reduction scheme [66] for the action space can be considered. Other approach for a huge number of training data is to reduce the number of training data. One of approaches is to use a sparse Gaussian process regression model (e.g. [67, 68]) as used in Saal et al. [69]. Another approach is to select the dominant training data for the modeling. We have proposed a subset selection method called Sparse Greedy Quadratic Minimization (SGQM) for multi-dimensional problems [70], originally proposed in [71] for one-dimensional problem. The fundamental concept is to sequentially select a data point from the training set in a greedy manner to minimize the approximation error of the maximum a posteriori probability (MAP) estimate of output y. Algorithm 3 shows the selection algorithm (see Appendix E for the definition of functions), and it has been successfully applied to the sequential intention estimation for intelligent driving assistance. To apply this approach would also be our future work. Moreover, we may use the information from two or more sensors in the feature extraction as treated in e.g. [54, 39]. Such an extension of our method to multimodal sensors (e.g. image and sound sensors) will also be addressed in the near future.

Algorithm 3: Sparse Greedy Quadratic Minimization for Multi-dimensional GP Regression

```
input  : Training data sets $\mathcal{D}_a$ and hyperparameter sets $\gamma_a$ for
         $a = \{1, 2, \ldots, d_y\}$, Precision $p$ or Maximum size of subset $\eta$, Size
         of randomized subset $\kappa$
output: Set of indices $\mathcal{S}$

Initialize index sets $\mathcal{I}, \mathcal{I}^* = \{1, 2, \ldots, N\}$; $\mathcal{S}, \mathcal{S}^* = \phi$
Set $\mathbf{P} := [], \mathbf{P}^* := []$.
repeat
    Choose $\kappa$ elements randomly from index sets $\mathcal{M} \subseteq \mathcal{I}, \mathcal{M}^* \subseteq \mathcal{I}^*$.
    Find $\arg\min_{i \in \mathcal{M}} \sum_a Q_a([\mathbf{P}, \mathbf{e}_i]\boldsymbol{\chi}_a([\mathbf{P}, \mathbf{e}_i]))$.
    Find $\arg\min_{i^* \in \mathcal{M}^*} \sum_a Q_a^*([\mathbf{P}^*, \mathbf{e}_{i^*}]\boldsymbol{\chi}_a([\mathbf{P}^*, \mathbf{e}_{i^*}]))$.
    Move $i$ from $\mathcal{I}$ to $\mathcal{S}$, $i^*$ from $\mathcal{I}^*$ to $\mathcal{S}^*$.
    Set $\mathbf{P} := [\mathbf{P}, \mathbf{e}_i]$, $\mathbf{P}^* := [\mathbf{P}^*, \mathbf{e}_{i^*}]$.
until $C_1(\mathbf{P}, \mathbf{P}^*) \leq \frac{p}{2} C_2(\mathbf{P}, \mathbf{P}^*)$ or $size(\mathcal{S}) = \eta$
```

Chapter 6

Conclusion

6.1. Summary

In this dissertation, we have proposed task-relevant manifold learning methods. The manifold learning methods are successfully applied in many applications [13, 14]. However, the obtained low-dimensional representation is sometimes not suitable for modeling. Based on the existing manifold learning methods, we have first proposed the input-output manifold learning with transfer function model (IOMLTF), which is a system identification method using the high-dimensional data (Chapter 3). The IOMLTF considers the fitting error to the dynamics represented using the transfer function models for the criterion of the manifold learning. The input-output manifold learning is achieved by solving the quadratic programming problem with quadratic constraints. It can be regarded as a multieigenvalue problem when the system is a single-input single-output system. In contrast with (original) Isomap, this solver seems to be natural since we solve two manifold learning problems simultaneously. Next, to capture the transient response, we have proposed the input-output manifold learning with state space model (IOMLSS), which is an extension of IOMLTF (Chapter 4). The extension is done by replacing the fitting error to the state space model version, and we show that the fitting error can be reformulated to the quadratic form. Then, an algorithm similar to IOMLTF can be applied to obtain the low-dimensional data and dynamics. Meanwhile, a partial manifold learning method is proposed. We consider two-factor models whose one factor is known and the other is unknown. Considering the effect of the factors to the observation, the unknown factor can be obtained suitably by the method. The method is applied to the object recognition tasks in order to obtain a suitable object parameter. We call the proposed method object manifold learning in this particular application. It allows us to simultaneously obtain a probabilistic sensor model that determines the informativeness of the exploratory action. In addition, to plan and execute the informative and compliant exploratory action, information maximization control that solves the problem in the optimal control framework is proposed. The proposed methods are validated through numerical simulations and experiments with actual robot hand. The results in the chapters show the effectiveness of proposed task-relevant manifold learning methods.

6.2. Future work

The effectiveness of our proposition is shown through the linear system modeling and robot perception. However, the experiments have limitations. The actual task with intelligent systems will be addressed. The sensor modalities addressed in this dissertation are visual and tactile. The intelligent systems consist of other modalities such as auditory or proximity. To treat or combine these sensor data, the previous studies with manifold learning, such as the sensor data fusion [72] or video sequence prediction [17], would be useful.

6.3. Future prospects

In the upcoming age of big data, the sensor data obtained from the intelligent systems would be huge, and more complicated phenomena can be captured by such a system. In order to tackle the modeling with such data, introducing deep generative models [73, 74] will be interesting. Moreover, we aim to determine a metric that is suitable for modeling. In this dissertation, we consider to add some penalty terms or constraints to the evaluation function for realization of the methods. Alternative to this realization, a new metric could be used instead of graph distances (Chapter 3 and 4) or Mahalanobis' generalized distance (Chapter 5). The previous studies [75, 76] will be useful for the realization.

Appendix

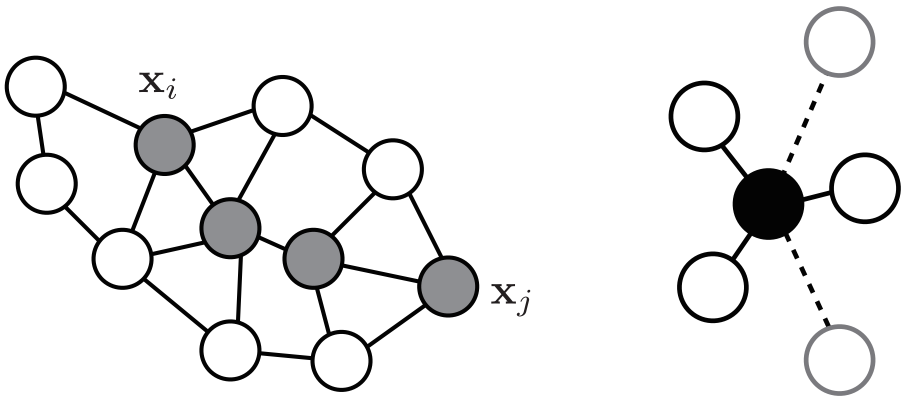
A. Approximation of the geodesic distance using graph

In this section, we describe how graph distance is computed as approximation of the geodesic distance from the data. Let us assume that the dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ is available. The basic idea is that the Geodesic distance between the sample is approximately equal to the Euclid distance in the neighborhood of the sample.

The Euclid distance among the samples can be calculated in a simple way, and we can consider the k-nearest neighbor (k-NN) graph of samples as shown in Fig. 6.1(a) based on the distance. To explain more concretely, let us consider the three nearest neighbors ($k = 3$) of the black node shown in Fig. 6.1(b). For the black node, five nodes are connected in total. Then, we can obtain the Euclid distance for the five pair of the black node. In the three nearest neighbors method, we cut the edges connected to the node except of the three nearest neighbors: the edges shown using the dotted line are cut. Repeating this for all nodes, we can construct the neighborhood graph like Fig. 6.1(a). By setting the length of the edge to Euclid distance between the nodes, the graph distance can be computed by solving the shortest path problem.

B. Multivariate eigenvalue problem

In this section, the setting of the multivariate eigenvalue problem (MEP) is described based on [18]. Additionally, the Horst-Jacobi method which is one of solvers for the MEP is shown.



(a) Example of the neighborhood graph (b) Example of k-NN ($k = 3$)

Figure A.1. Example of the k-NN graph.

Let us assume that a $n \times n$ positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and a n-dimensional column vector $\boldsymbol{x} \in \mathbb{R}^n$ are given. Further, a positive integer set,

$$\mathcal{P}_m = \{n_1, n_2, \ldots, n_m\}, \qquad \sum_{i=1}^{m} n_i = n$$

is assumed to be given. Then, we decompose $\mathbf{A}$ and $\mathbf{x}$ as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1m} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{m1} & \mathbf{A}_{m2} & \cdots & \mathbf{A}_{mm} \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} & \cdots & \mathbf{x}_m^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

where each matrix and vector is defined as $\mathbf{A}_{ij} \in \mathbb{R}^{n_i \times n_j}$, $\mathbf{x}_i \in \mathbb{R}^{n_i}$ respectively. Define

$$\boldsymbol{\Lambda} := \text{block diag}\{\lambda_1 \mathbf{I}_{n_1}, \cdots, \lambda_m \mathbf{I}_{n_m}\},$$

using $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$, then, the multivariate eigenvalue problem is to find a set $(\boldsymbol{\Lambda}, \boldsymbol{x})$ for $\mathcal{P}_m$.

$$\begin{cases} \mathbf{A}\mathbf{x} = \boldsymbol{\Lambda}\mathbf{x} \\ \|\mathbf{x}_i\|_2 = 1, \quad \mathbf{x}_i \in \mathbb{R}^{n_i} \\ (i = 1, 2, \cdots, m) \end{cases} \tag{B.1}$$

Algorithm 4: The Horst-Jacobi method

```
input  : matrix $\mathbf{A} \in \mathbb{R}^n$, initial column vector $\mathbf{x}^{(0)} \in \mathbb{R}^n$, a set $\mathcal{P}_m$
output: eigenvectors $\{\mathbf{x}_i\}_{i=1}^m$ and eigenvalues $\{\lambda_i\}_{i=1}^m$ corresponding to the
        (local) maximum value of $\sum_{i=1}^m \lambda_i$
for $k = 0, 1, \ldots,$ do
    for $i = 1, 2, \ldots, m$ do
        $\mathbf{y}_i^{(k)} := \sum_{j=1}^m \mathbf{A}_{ij}\mathbf{x}_j^{(k)}$
        $\lambda_i^{(k)} := \|\mathbf{y}_i^{(k)}\|_2$
        $\mathbf{x}_i^{(k+1)} := \frac{\mathbf{y}_i^{(k)}}{\lambda_i^{(k)}}$
    end
end
```

It is known that there are $\prod_{i=1}^m (2n_i)$ solutions.

Let us consider the problem to find the set $(\mathbf{\Lambda}, \boldsymbol{x})$ so that $\sum_{i=1}^m \lambda_i$ is maximized. Then, the problem of Eq. (B.1) can be represented as the following optimization problem.

$$\begin{cases} \text{Maximize} & r(\mathbf{x}) := \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} \\ \text{subject to} & \|\mathbf{x}_i\|_2 = 1, \quad \mathbf{x}_i \in \mathbb{R}^{n_i} \\ & (i = 1, 2, \cdots, m) \end{cases}$$

As a local solver for this optimization problem, the Horst-Jacobi method shown in Algorithm 4 is known.

C. Prediction error criterion in 4SID

Ignoring the noise term, the output predictions for the system Σ with given input sequence $\{\mathbf{u}_h, \ldots, \mathbf{u}_{T-1}\}$ is given as

$$\hat{\mathbf{Y}}_f = \mathbf{\Gamma}_h \mathbf{X}_f + \mathbf{H}_h^d \mathbf{U}_f,$$

where $\mathbf{X}_f = \mathbf{X}_{h|2h-1}$ is a Hankel matrix associated with the state sequences, and

$$\mathbf{\Gamma}_h = \begin{bmatrix} \mathbf{C} \\ \vdots \\ \mathbf{C}\mathbf{A}^{h-1} \end{bmatrix}$$

is called the extended observation matrix. The matrix,

$$\mathbf{H}_h^d = \begin{bmatrix} \mathbf{H}_0 & & & \\ \mathbf{H}_1 & \mathbf{H}_0 & & \\ \vdots & \vdots & \ddots & \\ \mathbf{H}_{h-1} & \mathbf{H}_{h-2} & \cdots & \mathbf{H}_0 \end{bmatrix},$$

is associated with the matrices

$$\mathbf{H}_i = \begin{cases} \mathbf{D} & (i = 0), \\ \mathbf{C}\mathbf{A}^{i-1}\mathbf{B} & (i > 0), \end{cases}$$

called Markov parameters. The term $\mathbf{\Gamma}_h\mathbf{X}_f$ is used even though both matrices are unknown. However, it can be estimated by the following oblique projection,

$$\mathbf{Y}_f \underset{\mathbf{U}_f}{/} \mathbf{W}_p = \mathbf{\Gamma}_h\hat{\mathbf{X}}_f. \tag{C.1}$$

A property of the oblique projection indicates that a matrix $\mathbf{L}_w$ which satisfies the following relationship exists,

$$\mathbf{Y}_f \underset{\mathbf{U}_f}{/} \mathbf{W}_p = \mathbf{L}_w\mathbf{W}_p, \tag{C.2}$$

and finally $\hat{\mathbf{Y}}_f$ is obtained as follows,

$$\hat{\mathbf{Y}}_f = \mathbf{L}_w\mathbf{W}_p + \mathbf{H}_h^d\mathbf{U}_f. \tag{C.3}$$

From the relationship, it is obvious that updating the $\mathbf{L}_w$ corresponds to the update of the extended observability matrix and state vector. See [28] for the details.

D. Definition of the vectors and matrices for evaluation of mutual information

Φ_{ab} which is the (a,b)-th element of $\boldsymbol{\Phi}(\mathbf{x},\mathbf{x}') \in \mathbb{R}^{d_\mathbf{y}\times d_\mathbf{y}}$, and $\mathbf{C}(\mathbf{x}) \in \mathbb{R}^{d_{\boldsymbol{\theta}}\times d_\mathbf{y}}$ are defined as follows:

$$\begin{aligned}
\Phi_{ab} &= \boldsymbol{\beta}_a^{\mathrm{T}}\boldsymbol{\Lambda}_{ab}(\mathbf{x},\mathbf{x}')\boldsymbol{\beta}_b - m_a(\mathbf{x})m_b(\mathbf{x}') \\
&\quad + \delta_{\mathbf{x}\mathbf{x}'}\delta_{ab}\Big(\alpha_a^2 - \mathrm{Tr}\big((\mathbf{K}_a+\sigma_a^2\mathbf{I})^{-1}\boldsymbol{\Lambda}_{aa}(\mathbf{x},\mathbf{x}')\big)\Big), \\
\mathbf{C} &= \boldsymbol{\Psi}(\mathbf{x}) - \boldsymbol{\mu}\mathbf{m}(\mathbf{x})^{\mathrm{T}}.
\end{aligned}$$

The a-th entry of $\mathbf{m}(\mathbf{x}) \in \mathbb{R}^{d_\mathbf{y}}$ is $m_a = \boldsymbol{\beta}_a^{\mathrm{T}}\boldsymbol{\lambda}_a(\mathbf{x})$, and $\boldsymbol{\beta}_a$ is defined as $\boldsymbol{\beta}_a = (\mathbf{K}_a+\sigma_a^2\mathbf{I})^{-1}\mathbf{y}_a \in \mathbb{R}^N$.

The i-th entry of the vector $\boldsymbol{\eta}_a$ for $i = 1,2,\ldots,N$ is

$$\begin{aligned}
\eta_{ai} &= \alpha_a^2 \det\Big(\boldsymbol{\Sigma}\big(\mathbf{H}_a^\theta\big)^{-1} + \mathbf{I}\Big)^{-\frac{1}{2}} \\
&\quad \times \exp\Big(-\frac{1}{2}\big(\boldsymbol{\mu}-\boldsymbol{\theta}^{(i)}\big)^{\mathrm{T}}\big(\boldsymbol{\Sigma}+\mathbf{H}_a^\theta\big)^{-1}\big(\boldsymbol{\mu}-\boldsymbol{\theta}^{(i)}\big)\Big) \\
&\quad \times \exp\Big(-\frac{1}{2}\big(\mathbf{x}-\boldsymbol{x}^{(i)}\big)^{\mathrm{T}}\big(\mathbf{H}_a^\mathbf{x}\big)^{-1}\big(\mathbf{x}-\mathbf{x}^{(i)}\big)\Big).
\end{aligned}$$

The (i,j)-th entry of the matrix $\boldsymbol{\Lambda}_{ab}$ is represented as follows:

$$\begin{aligned}
\Lambda_{ab,ij} &= \alpha_a^2\alpha_b^2 \det\bigg(\Big(\big(\mathbf{H}_a^\theta\big)^{-1} + \big(\mathbf{H}_b^\theta\big)^{-1}\Big)\boldsymbol{\Sigma} + \mathbf{I}\bigg)^{-\frac{1}{2}} \\
&\quad \times \exp\Big(-\frac{1}{2}\big(\boldsymbol{\theta}^{(i)}-\boldsymbol{\theta}^{(j)}\big)^{\mathrm{T}}\big(\mathbf{H}_{ab}^\theta\big)^{-1}\big(\boldsymbol{\theta}^{(i)}-\boldsymbol{\theta}^{(j)}\big)\Big) \\
&\quad \times \exp\Big(-\frac{1}{2}\big(\boldsymbol{\theta}_{ab}^{ij}-\boldsymbol{\mu}\big)^{\mathrm{T}}\mathbf{R}_{ab}^{-1}\big(\boldsymbol{\theta}_{ab}^{ij}-\boldsymbol{\mu}\big)\Big) \\
&\quad \times \exp\Big(-\frac{1}{2}\big(\mathbf{x}-\mathbf{x}^{(i)}\big)^{\mathrm{T}}\big(\mathbf{H}_a^x\big)^{-1}\big(\mathbf{x}-\mathbf{x}^{(i)}\big)\Big) \\
&\quad \times \exp\Big(-\frac{1}{2}\big(\mathbf{x}'-\mathbf{x}^{(j)}\big)^{\mathrm{T}}\big(\mathbf{H}_b^x\big)^{-1}\big(\mathbf{x}'-\mathbf{x}^{(j)}\big)\Big),
\end{aligned}$$

where $\mathbf{H}_{ab}^\theta = \mathbf{H}_a^\theta + \mathbf{H}_b^\theta$ and

$$\begin{aligned}
\boldsymbol{\theta}_{ab}^{ij} &= \mathbf{H}_b^\theta\big(\mathbf{H}_{ab}^\theta\big)^{-1}\boldsymbol{\theta}^{(i)} + \mathbf{H}_a^\theta\big(\mathbf{H}_{ab}^\theta\big)^{-1}\boldsymbol{\theta}^{(j)}, \\
\mathbf{R}_{ab} &= \Big(\big(\mathbf{H}_a^\theta\big)^{-1} + \big(\mathbf{H}_b^\theta\big)^{-1}\Big)^{-1} + \boldsymbol{\Sigma},
\end{aligned}$$

are defined. The vector $\boldsymbol{\psi}_a$, which is the a-th column of $\boldsymbol{\Psi}$ for $a = 1, 2, \ldots, d_{\mathbf{y}}$ is defined as follows:

$$\boldsymbol{\psi}_a = \sum_{i=1}^{N} \beta_{ai}\eta_{ai}(\mathbf{x}) \left(\left(\mathbf{H}_a^{\theta}\right)^{-1} + \boldsymbol{\Sigma}^{-1} \right)^{-1} \left(\left(\mathbf{H}_a^{\theta}\right)^{-1} \boldsymbol{\theta}^{(i)} + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \right),$$

where β_{ai} is the i-th element of $\boldsymbol{\beta}_a$.

E. Definition of the functions for subset selection of the training data

Functions $Q_a, Q_a^*, \boldsymbol{\chi}_a, C_1, C_2$ in Algorithm 3 are defined as follows:

$$\begin{aligned}
Q_a(\boldsymbol{\alpha}) &= -\mathbf{y}_a^{\mathrm{T}}\mathbf{K}_a\boldsymbol{\alpha} + \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}(\sigma_a^2\mathbf{K}_a + \mathbf{K}_a^{\mathrm{T}}\mathbf{K}_a)\boldsymbol{\alpha}, \\
Q_a^*(\boldsymbol{\alpha}) &= -\mathbf{y}_a^{\mathrm{T}}\boldsymbol{\alpha} + \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}(\sigma_a^2\mathbf{I} + \mathbf{K}_a)\boldsymbol{\alpha}, \\
\boldsymbol{\chi}_a(\mathbf{P}) &= \left(\mathbf{P}^{\mathrm{T}}(\sigma_a^2\mathbf{K}_a + \mathbf{K}_a^{\mathrm{T}}\mathbf{K}_a)\mathbf{P}\right)^{-1}\mathbf{P}^{\mathrm{T}}\mathbf{K}_a^{\mathrm{T}}\mathbf{y}_a, \\
C_1(\mathbf{P}, \mathbf{P}^*) &= \sum_{a=1}^{d_{\mathbf{y}}} \Big(Q_a(\mathbf{P}\boldsymbol{\chi}_a(\mathbf{P})) + \sigma_a^2 Q_a^*(\mathbf{P}^*\boldsymbol{\chi}_a(\mathbf{P}^*)) \\
&\qquad\qquad + \tfrac{1}{2}\|\mathbf{y}_a\|^2 \Big), \\
C_2(\mathbf{P}, \mathbf{P}^*) &= \sum_{a=1}^{d_{\mathbf{y}}} \Big(|Q_a(\mathbf{P}\boldsymbol{\chi}_a(\mathbf{P}))| \\
&\qquad + \left| \sigma_a^2 Q_a^*(\mathbf{P}^*\boldsymbol{\chi}_a(\mathbf{P}^*)) + \tfrac{1}{2}\|\mathbf{y}_a\|^2 \right| \Big),
\end{aligned}$$

and $\mathbf{e}_i$ refers to the i-th column vector of $N \times N$ identity matrix.

Bibliography

[1] H. Saal, J.-A. Ting, and S. Vijayakumar, “Active sequential learning with tactile feedback,” in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pp. 677–684, 2010.

[2] N. Lawrence, “Probabilistic non-linear principal component analysis with Gaussian process latent variable models,” *Journal of Machine Learning Research*, vol. 6, pp. 1783–1816, 2005.

[3] H. Hotelling, “Analysis of a complex of statistical variables into principal components.,” *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[4] W. S. Torgerson, “Multidimensional scaling: I. theory and method,” *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[5] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[6] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[7] S. T. Roweis and L. K. Saul, “Nonlinear Dimensionality Reduction by Locally Linear Embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[8] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[9] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction.* Springer Science & Business Media, 2007.

[10] L. J. van der Maaten, E. O. Postma, and H. J. van den Herik, "Dimensionality reduction: A comparative review," *Journal of Machine Learning Research*, vol. 10, no. 1-41, pp. 66–71, 2009.

[11] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on Computer vision*, vol. 2, pp. 1150–1157, 1999.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.

[13] R. Pless, "Image spaces and video trajectories: using Isomap to explore video sequences," in *Proceedings of the 9th IEEE International Conference on Computer Vision*, pp. 1433–1440, 2003.

[14] A. Shon, K. Grochow, A. Hertzmann, and R. P. Rao, "Learning shared latent structure for image synthesis and robotic imitation," in *Advances in Neural Information Processing Systems*, pp. 1233–1240, 2005.

[15] H. Ohlsson, J. Roll, and L. Ljung, "Manifold-constrained regressors in system identification," in *47th IEEE Conference on Decision and Control*, pp. 1364–1369, 2008.

[16] J. Wang, A. Hertzmann, and D. M. Blei, "Gaussian process dynamical models," in *Advances in neural information processing systems*, pp. 1441–1448, 2005.

[17] A. Damianou, M. K. Titsias, and N. D. Lawrence, "Variational Gaussian process dynamical systems," in *Advances in Neural Information Processing Systems*, pp. 2510–2518, 2011.

[18] M. T. Chu and J. L. Watterson, "Multivariate eigenvalue problem: I. algebraic theory and power method," *SIAM Journal on Scientific Computing*, vol. 14, no. 5, pp. 1089–1106, 1993.

[19] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-100)," tech. rep., Columbia University, 1996.

[20] V. Verdult, J. A. K. Suykens, J. Boets, I. Goethals, and B. D. Moor, "Least Squares Support Vector Machines for Kernel CCA in Nonlinear State-Space Identification," in *Proceedings of the 16th international symposium on Mathematical Theory of Networks and Systems (MTNS2004)*, pp. 1–11, 2004.

[21] L. Ljung, *System Identification: Theory for the User.* Prentice Hall, 1999.

[22] Y. Zhu, "Estimation of an N-L-N Hammerstein-Wiener model," *Automatica*, vol. 38, no. 9, pp. 1607 – 1614, 2002.

[23] E.-W. Bai, "An optimal two-stage identification algorithm for Hammerstein-Wiener nonlinear systems," *Automatica*, vol. 34, no. 3, pp. 333–338, 1998.

[24] P. Crama and J. Schoukens, "Hammerstein-Wiener system estimator initialization," *Automatica*, vol. 40, no. 9, pp. 1543 – 1550, 2004.

[25] L.-H. Zhang, L.-Z. Liao, and L.-M. Sun, "Towards the global solution of the maximal correlation problem," *Journal of Global Optimization*, vol. 49, pp. 91–107, 2011.

[26] T. Katayama, *Subspace Methods for System Identification.* Springer, 2005.

[27] P. V. Overschee and B. D. Moor, "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, no. 1, pp. 75–93, 1994.

[28] P. Trnka and V. Havlena, "Subspace identification as multi-step predictions optimization," in *Proceedings of the 5th IASTED International Conference on Modelling, Simulation and Optimization*, pp. 223–228, 2005.

[29] P. Trnka, *Subspace Identification Methods.* PhD thesis, Czech Technical University in Prague, 2007.

[30] D. Tanaka, T. Matsubara, and K. Sugimoto, "System identification with input-output manifold learning," *The IEICE Transactions on Fundamentals*

of Electronics, Communications and Computer Sciences (Japanese Edition), vol. J96-A, no. 8, pp. 551–561, 2013. (in Japanese).

[31] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[32] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[33] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Multifactor Gaussian process models for style-content separation," in *Proceedings of the 24th Annual International Conference on Machine Learning*, pp. 975–982, 2007.

[34] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.* MIT Press, 2006.

[35] K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Task-driven tactile exploration," in *Proceedings of Robotics Science and Systems*, 2010.

[36] B. Browatzki, V. Tikhanoff, G. Metta, H. Bulthoff, and C. Wallraven, "Active object recognition on a humanoid robot," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2021–2028, 2012.

[37] N. Lepora, U. Martinez-Hernandez, and T. Prescott, "Active touch for robust perception under position uncertainty," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 3020–3025, 2013.

[38] A. Schneider, J. Sturm, C. Stachniss, M. Reisert, H. Burkhardt, and W. Burgard, "Object identification with tactile sensors using bag-of-features," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 243–248, 2009.

[39] J. Sinapov, C. Schenck, K. Staley, V. Sukhoy, and A. Stoytchev, "Grounding semantic categories in behavioral interactions: Experiments with 100 objects," *Robotics and Autonomous Systems*, vol. 62, no. 5, pp. 632–645, 2014.

[40] J. A. Fishel and G. E. Loeb, "Bayesian exploration for intelligent identification of textures," *Frontiers in Neurorobotics*, vol. 6, no. 4, 2012.

[41] C. E. Rasmussen, *Gaussian processes for machine learning.* MIT Press, 2006.

[42] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[43] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bülthoff, and C. Wallraven, "Active object recognition on a humanoid robot," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2021–2028, 2012.

[44] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[45] M. Pontil and A. Verri, "Support vector machines for 3d object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 637–646, 1998.

[46] J. Sinapov, M. Wiemer, and A. Stoytchev, "Interactive learning of the acoustic properties of household objects," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2518–2524, 2009.

[47] E. Torres-Jara, L. Natale, and P. Fitzpatrick, "Tapping into touch," in *5th International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, vol. 123, pp. 79–86, 2005.

[48] P. Dallaire, P. Giguère, D. Émond, and B. Chaib-draa, "Autonomous tactile perception: A combined improved sensing and bayesian nonparametric approach," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 422–435, 2014.

[49] A. M. Okamura and M. R. Cutkosky, "Feature detection for haptic exploration with robotic fingers," *The International Journal of Robotics Research*, vol. 20, no. 12, pp. 925–938, 2001.

[50] S. Chitta, M. Piccoli, and J. Sturm, "Tactile object class and internal state recognition for mobile manipulation," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2342–2348, 2010.

[51] G. Heidemann and M. Schopfer, "Dynamic tactile sensing for object identification," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 813–818, April 2004.

[52] M. Johnsson and C. Balkenius, "Experiments with proprioception in a self-organizing system for haptic perception," in *Proceedings of Towards Autonomous Robotic Systems 2007*, pp. 239–245, 2007.

[53] N. Gorges, S. Navarro, D. Göger, and H. Worn, "Haptic object recognition using passive joints and haptic key features," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2349–2355, 2010.

[54] O. Kroemer, C. H. Lampert, and J. Peters, "Learning dynamic tactile sensing with robust vision-based training," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 545–557, 2011.

[55] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky., and H. F. Durrant-Whyte, "Information based adaptive robotic exploration," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 540–545, 2002.

[56] J. Le Ny and G. J. Pappas, "On trajectory optimization for active sensing in Gaussian process models," in *Proceedings of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pp. 6286–6292, 2009.

[57] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[58] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms.* New York, NY, USA: Cambridge University Press, 2002.

[59] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck, "Analytic moment-based Gaussian process filtering," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 225–232, 2009.

[60] D. P. Bertsekas, *Dynamic Programming and Optimal Control.* Athena Scientific, 2nd ed., 2000.

[61] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.," in *Proceedings of International Conference on Informatics in Control, Automation and Robotics*, pp. 222–229, 2004.

[62] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[63] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems 15*, pp. 1523–1530, Cambridge, MA: MIT Press, 2003.

[64] R. Storn and K. Price, "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[65] A. d'Avella, P. Saltiel, and E. Bizzi, "Combinations of muscle synergies in the construction of a natural motor behavior," *Nature neuroscience*, vol. 6, no. 3, pp. 300–308, 2003.

[66] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[67] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*, pp. 1257–1264, MIT Press, 2006.

[68] Y. Shen, M. Seeger, and A. Y. Ng, "Fast Gaussian process regression using KD-Trees," in *Advances in Neural Information Processing Systems 18*, pp. 1225–1232, MIT Press, 2006.

[69] H. P. Saal, J. Ting, and S. Vijayakumar, "Active estimation of object dynamics parameters with tactile sensors," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 916–921, 2010.

[70] T. Matsubara, J. V. Miró, D. Tanaka, J. Poon, and K. Sugimoto, "Sequential intention estimation of a mobility aid user driving behavior for intelligent driving assistance," in *Proceedings of 24th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 444–449, 2015.

[71] A. J. Smola and P. L. Bartlett, "Sparse greedy gaussian process regression," in *Advances in Neural Information Processing Systems 13*, pp. 619–625, MIT Press, 2001.

[72] M. A. Davenport, C. Hegde, M. F. Duarte, and R. G. Baraniuk, "Joint manifolds for data fusion," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2580–2594, 2010.

[73] A. C. Damianou and N. D. Lawrence, "Deep gaussian processes," in *Proceedings of the 16th International Workshop on Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[74] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[75] O. C. Jenkins and M. J. Matarić, "A spatio-temporal extension to isomap nonlinear dimension reduction," in *Proceedings of the 21st International Conference on Machine Learning*, vol. 69 of *ACM International Conference Proceeding Series*, ACM, 2004.

[76] T. Chen and L. Ljung, "Constructive statespace model induced kernels for regularized system identification," in *Proceedings of the 19th IFAC World Congress*, pp. 1047–1052, 2014.