Learning of Motion Generation for Various Situations based on Sequence-to-Sequence Models
（Sequence-to-Sequenceモデルに基づく多様な状況への動作生成の学習）

March 2020

Ph.D. dissertation

Course in Mathematics, Electronics, and Informatics,
Programs in Science and Engineering,
Graduate School of Science and Engineering,
Saitama University

Kyo Kutsuzawa

Supervised by Dr. Toshiaki Tsuji

Contents

Chapter 1

Introduction

1.1 Background

As of 2019, robots are primarily used in factory automation. They are used for manufacturing cars, electric and electronic devices, *etc.* They are efficient in high-reproducibility repetitive tasks.

Furthermore, their application is expected to extend to our daily tasks, such as cooking, cleaning, and other day-to-day household tasks. However, currently, robots are not efficient at such tasks. In addition, even in factory automation, robots have not yet been applied to tasks that handle various products called *high-mix low-volume production* and *various kinds and volumes production*.

1.2 Issue

Why are the current robots not efficient at these tasks? A primary reason is that these tasks are executed in diverse situations. For example, in cooking, items are organized differently in a refrigerator each time. In addition, robots should progress the task quickly or carefully based on user priority, such as speed or safety. Further, physical parameters, such as friction coefficients, are also different each time. If the situation is different, the task cannot be accomplished by the same commands generally. Therefore, robots require certain motion generators that can generate commands based on varied situations.

The motion generators should receive concrete representations of situations and generate commands based on the situations. One of the common representations of situations is the current state; it includes various information about disturbances, changes in physical parameters, *etc.* To process them, the motion generators should receive the current state online. In addition, some kinds of situations such as limitations, goals of the tasks, and user priority should be provided externally. In this dissertation, such information is referred as an *external order*. In summary, the motion generation is illustrated in Fig. 1.1. Here, the situations indicate what the motion generators receive: the current states and the external orders.

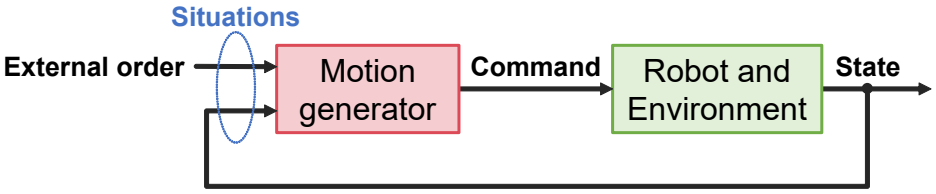


Figure 1.1: Overview of the motion generation.

Considering the diversity of situations in our daily tasks, the external order can be quite diverse. For example, there exist various limitations, such as velocity limits, torque limits, and working ranges. Furthermore, there are various user orders, such as fast, slow, and careful. Moreover, the representation of the external order vary despite a same intended instruction. For example, the most straightforward way in representing a target position is the coordinate. However, there exists another way of representation; in the reaching task, the position of the target object can be specified by an RGB-D image.

It is considered that there are two aspects of the diversity of external orders—values and representations. The former means that there are external orders with the same form that have different values. For example, when the velocity limit is represented by using a threshold value, its value can be both low and high. The latter refers that there are different representations in external orders with the same meaning. For example, in representing a target position, it can be both coordinates and RGB-D images. Also, in representing user orders, there are various ways—voices, texts, one-hot vectors, or its continuous embeddings.

Thus, the external orders vary in both the values and representations. Therefore, the underlying issue is, *how will motion generators process various external orders?*

1.3 Approach

In this section, the motion generators that are suitable for solving the aforementioned issue are discussed. There exist two main approaches to motion generation—approaches based on mathematical optimization and machine learning. In the approach based on mathematical optimization, external orders are expressed mathematically such as in objective functions. This approach solves the motion generation problems mathematically. For example, model predictive control [1], dynamic programming [2, 3], and sampling-based motion planning methods, such as rapidly-exploring random tree [4], are well-known mathematical methods. Such methods, however, require mathematical expressions. They cannot handle numerical expressions such as human demonstrations.

In contrast, the approach based on machine learning can handle numerical expressions. Using a set of pairs of external orders and corresponding trajectories, such as human demonstrations, motion generators can be trained to associate with them. Moreover, they can imitate mathematical optimization methods [5, 6]. However, this approach has an

issue in dealing with various external orders. Generally, trained models cannot process inputs that significantly deviate from training data, *i.e.*, trained models can handle only limited external orders. Considering the diverse external orders as discussed earlier, it is difficult to input all possible external orders to the models.

This issue of the approach based on machine learning is caused by the training method that provides external orders directly. Therefore, the issue can be resolved by training the motion generators without external orders. This can be achieved by using certain intermediate representation—by learning the relationship between such intermediate representation and commands, the motion generators become independent of the external orders. Then, the external orders are associated with the intermediate representation after training the motion generators. The use of the intermediate representation resolves the issue of the diversity of representations in the external orders. Note that while this method can considerably reduce the limitations on the external orders, the motion generators come to depend on the representation ability of the intermediate representation instead, as discussed in Chapter 1.5.

Hence, *latent representation* of trajectories can be used as an intermediate representation. This latent representation expresses features that can uniquely specify a single trajectory in a set of trajectories. Although this latent representation cannot be directly observed, it can be obtained by using a set of trajectories. Because the latent representation depends only on trajectories, it enables the motion generators to not depend on the external orders.

Recently, the use of latent representations has been studied [7–13]. However, for the aforementioned motion generation, there exist certain requirements for motion generators and latent representations. First, motion generators should be used online. The motion generators should work online to deal with certain situations, such as disturbances, that are represented in the current state. Second, the latent representations should represent entire trajectories. This is because external orders often determine the entire trajectory. Considering the aforementioned requirements, most conventional works on the latent representations of motions [7–10] are not suitable to address the issue discussed in this study. Although certain studies [11–13] consider latent representations of short-term trajectories, they did not study the changes in situations, as discussed earlier.

In this study, I use encoder-decoder models for time-series, which are also known as *sequence-to-sequence (seq2seq) models* [14, 15]. Seq2seq models can extract latent representations from trajectories and generate motions online. Therefore, it is expected that motion generators for various situations can be realized using seq2seq models.

1.4 Seq2seq Model

Seq2seq models are neural network architecture used to convert an input sequence to an output sequence. A seq2seq model is composed of two recurrent neural networks (RNNs). The input-side RNN is called an *encoder*. It receives input trajectories and

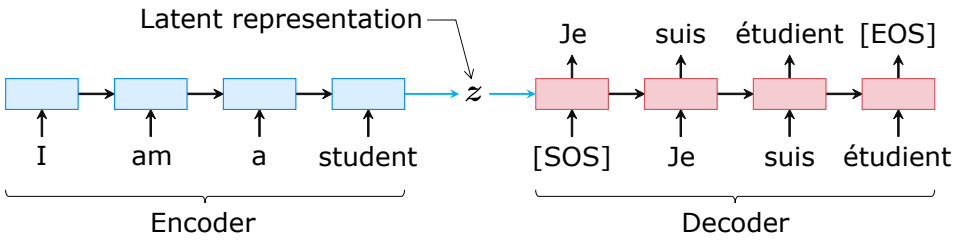


Figure 1.2: Sequence-to-sequence model. This model translates an English sentence "I am a student" to an French sentence "Je suis étudiant." The blue block denotes an encoder RNN and the orange block denotes a decoder RNN. The blue arrow is an fixed dimensional vector which contains internal representation. [SOS] and [EOS] are special terms meaning start-of-sequence and end-of-sequence, respectively.

converts them to latent representations. The output-side RNN is called the *decoder*. It receives the latent representations and generates output trajectories. The decoder can be used as a motion generator that generates commands based on current states and latent representations.

Seq2seq models are applied to many kinds of tasks such as machine translation [14,15], conversational modeling [16], and summarizing [17]. Most applications are in natural language processing. Certain studies have recently applied seq2seq models to robotics. Yamada *et al.* used seq2seq models in robots to learn the relationship between linguistic commands and appropriate actions [18]. Kutsuzawa *et al.* proposed a method of chunking a trajectory into groups of samples to solve an issue caused by lengthy trajectories of robots [19].

The converting process in machine translation is illustrated in Fig. 1.2. First, the encoder receives a source sentence word-by-word. It converts the received words to latent representation internal representation and memorizes them into internal memories. To memorize the received words during this process, long short-term memories (LSTMs) [20, 21] and gated recurrent units (GRUs) [14] are often used. The meaning of the source sentence is reflected in the latent representation. Then, the encoder provides the latent representation to the decoder. The decoder receives a special token representing the start of a sentence (SOS) and generates the first word of the translated sentence according to the latent representation. Then, the decoder receives the first word, generates the next word, and repeats the process. Finally, the decoding process is complete when the decoder generates a special token representing the end of the sentence (EOS).

1.5 Research Topics

As discussed earlier, seq2seq models can be an efficient approach for understanding how the motion generators would process various external orders. However, there are certain topics to be discussed.

The first topic is training methods for various trajectories. Although the use of the latent representations enables training of trajectories independent of certain external orders, there exist limitations for acceptable external orders. All models cannot generate trajectories that significantly deviate from the training data. Therefore, the diversity of training trajectories limits the performance of the models. Thus, training methods of various trajectories to the seq2seq models should be discussed.

It is difficult to train various trajectories to seq2seq models when there are constraints and complex dynamics. The randomly generated trajectories are not guaranteed to exhibit sufficient diversity and satisfy the dynamic conditions. For example, when subject to dynamic constraints, most random trajectories do not succeed in the task of turning over pancakes upside down within a realistic range of motion. Therefore, it is difficult to obtain a repertory of trajectories that exhibit sufficient diversity.

The second topic is to associate certain external orders with latent representations. As the seq2seq models do not specify what the latent representations represent, they demonstrate low interpretability. Therefore, associating them manually is difficult and the association methods are not straightforward.

In addition, the robustness of the decoders of seq2seq models in online motion generation is also an important topic. Owing to the online motion generation, the motion generators may detect changes in the environment and compensate them to follow the trajectories represented by the latent representations. However, it is not confirmed if the trained decoders have such robustness. Therefore, the robustness of decoders should be validated.

Thus, this study aims to address the aforementioned issues.

1.6 Construction of this Dissertation

An overview of the construction of this dissertation is illustrated in Fig. 1.3. Note that the contents in Chapter 2 were already presented by the author as the master's course thesis [22]. However, it is included in this dissertation as it is one of the essential parts of this research.

First, trajectory deformation-based training methods of seq2seq models were proposed. The seq2seq models are trained to deform given trajectories to satisfy the constraints and complex dynamics. Using the proposed methods, users can train the seq2seq models by preparing various input trajectories regardless of the environment. Then, the seq2seq models are trained using curriculum learning along with penalties. Finally, users can train various valid trajectories to the seq2seq models and control the diversity of the trajectories. These methods solely require models with constraints and environmental parameters. Chapter 2 discusses the dynamic constraint. Chapter 3 discusses the discontinuous environment.

In addition, Chapter 3 discusses the robustness of the seq2seq models against fluctuations in environmental parameters was evaluated. Although the models are trained by

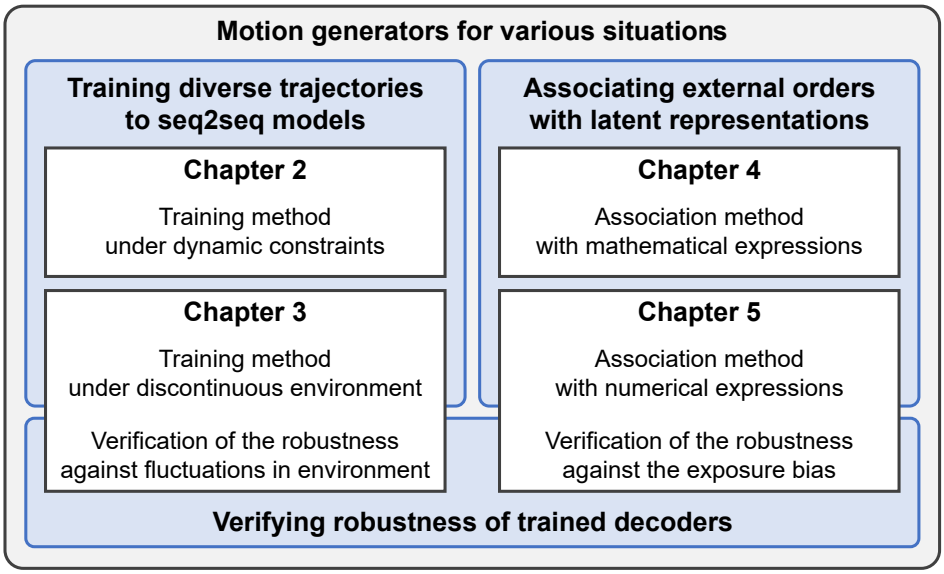


Figure 1.3: Overview of the construction of this dissertation.

the proposed method with only a single set of parameters, these trained models result in smaller reproduction errors than models trained by other methods.

Second, association methods of the latent representations with two types of external order representations were proposed—mathematical expressions and numerical expressions. Chapter 4 discusses the external orders that are represented as mathematical expressions, such as objective functions. To obtain latent representations that minimize the objective functions, an optimization method based on backpropagation [23] for the latent representations was proposed. In this chapter, several objective functions, different in both forms and values, were employed. Chapter 5 discusses the external orders that are represented as numerical data, such as human demonstrations. A training method for additional models to associate numerical external orders with appropriate latent representations was proposed.

Further, Chapter 5 evaluates the robustness of the motion generators against *the exposure bias* [24,25], which is caused by training of RNNs without environmental models [26]. Using the proposed method, the robustness of the motion generators can be improved without increasing the number of numerical external orders.

This study is summarized in Chapter 6. Although all chapters are associated with the main aim of this study, each chapter has a distinct background and conclusion.

Chapter 2

Training of Seq2seq Models under Dynamic Constraints[1][2]

In this chapter, training methods of various trajectories for seq2seq models is considered. When there exist dynamic constraints in a given task, it is difficult to obtain a repertory of trajectories that has enough diversity as well as satisfying the dynamic constraints.

This chapter takes dynamic manipulation as a typical example including dynamic constraints. A training method based on the idea of trajectory deformation is proposed.

2.1 Introduction

Object manipulation is one of the main topics in robotics. Grasping and carrying objects with grippers is one of the most popular issues, while there exist another type of manipulation called nonprehensile manipulation [2, 27], which does not involve grasping. Nonprehensile manipulation includes wide variety of motion such as pushing, throwing, rolling, batting, *etc.* Lynch *et al.* realized a robot performing a juggler's skill called butterfly [28]. Kormushev *et al.* realized a manipulator flipping a pancake on a flying pan by teaching from humans and reinforcement learning [29]. Bauza *et al.* realized a planar pushing motion using a variation in the Gaussian process [30]. Some studies address the motion planning of nonprehensile manipulation by dynamic programming [3, 31]. Neural networks have attracted attention in some studies recently [32–34].

Among nonprehensile manipulation, this chapter focuses on dynamic manipulation [35], which utilizes dynamic effects such as inertial force and gravity. Mason *et al.* defined the concept of dynamic closure as the use of inertial force to sustain contact constraint [35]. Dynamic closure is accomplished by applying appropriate acceleration to exert inertial

[1] The contents in this chapter were already presented by the author as the master's course thesis [22]; I included it since it is an essential part of this study.

[2] The contents in this chapter were also published in: K. Kutsuzawa, S. Sakaino, and T. Tsuji, "Sequence-to-sequence models for trajectory deformation of dynamic manipulation," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc.*, 2017, pp. 5227–5232.

force to sustain static friction. Tsuji *et al.* realized a manipulator turning over a pancake with a spatula [36]. In [36], appropriate acceleration is determined so that the resultant force of gravity and inertial force fits into the friction cone.

In dynamic manipulation, inertial force is used instead of grasping. That is realized by a robot inputting appropriate acceleration. This operation is executed by the same degrees of freedom as the motion. Therefore, applying appropriate acceleration causes motion of an object or a desired motion will not satisfy the dynamic closure. These two requirements can not be decoupled. Thus, trajectory planning in dynamic manipulation is a difficult issue. Previous methods of trajectory planning in dynamic manipulation need to search high dimensional state spaces [4, 37].

The issue is training methods of trajectories that have enough diversity as well as satisfying the dynamic constraints. This issue can be considered as a combination of two sub-issues—the diversity of trajectories and the dynamic constraints. It is relatively easy to obtain enough diversity of trajectories without regarding the dynamic constraints. In addition, expressing the conditions of dynamic constraints is also easy.

Therefore, it is considered that trajectory deformation is an effective approach for such an issue. This approach of trajectory deformation is taken to divide motion planning into two steps: a step of trajectory planning according to an requirement and another step of deforming based on another requirement. Kurniawati *et al.* proposed a method of deformation that a trajectory for collision avoidance is deformed to ensure feasibility for a dynamic model of a robot [38]. Lamiraux *et al.* proposed a method of trajectory planning for obstacle avoidance with two steps: planning obstacle avoidance trajectory based on prior information and then deform it to avoid unexpected obstacles in online [39]. Zegers *et al.* used neural networks to deform limit cycles to desired shape [40]. The deformation approach has also been applied to nonprehensile manipulation. Pekarovskiy *et al.* applied laplacian trajectory editing [41] to batting motion [42]. This method deforms a preset trajectory to pass through a given hitting point while avoiding obstacles. However, these methods are not been applied to dynamic manipulation.

By using a trajectory deformation method, trajectories can be obtained by planning desired trajectories and deforming them to satisfy dynamic constraint. To realize trajectory deformation for dynamic manipulation, obtaining deformation methods is an issue. Thus, this chapter aims to use seq2seq models to learn a method of trajectory deformation to satisfy dynamic constraint while sustaining the original shape. In addition, this chapter proposes an unsupervised method of training seq2seq models to learn a deformation method. Labeled data is not required thanks to unsupervised learning. Only unlabeled trajectories and mathematical representation of constraint are necessary.

This chapter is constructed with the following sections: Section sec:chapter2-method describes the proposed method. Main architecture of seq2seq models for trajectory deformation and a design method of learning curriculum are explained. Then, Section 2.3 shows implementation of the proposed method and confirms the validity of the proposed method by simulation and an experiment. Finally, this chapter is concluded in Section 2.4.

2.2 Method

2.2.1 Physics Model

This chapter considers a task of turning over a pancake with a spatula as illustrated in Fig. 2.1. The same task is considered in [36]. This chapter considers planar physics.

A state equation is expressed as follows:

$$\boldsymbol{x}[k+1] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k], \tag{2.1}$$

where

$$\boldsymbol{x}[k] = [y[k], z[k], \theta[k], v_y[k], v_z[k], v_\theta[k]]^\top, \tag{2.2}$$

$$\boldsymbol{u}[k] = [a_y[k], a_z[k], a_\theta[k]]^\top, \tag{2.3}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_3 & \Delta t \boldsymbol{I}_3 \\ \boldsymbol{O}_3 & \boldsymbol{I}_3 \end{bmatrix}, \tag{2.4}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{\Delta t^2}{2} \boldsymbol{I}_3 \\ \Delta t \boldsymbol{I}_3 \end{bmatrix}. \tag{2.5}$$

Here, $\boldsymbol{x}[k]$ is a state variable at k-th sample, $\boldsymbol{u}[k]$ is an acceleration input at k-th sample, Δt is a sampling interval, $\boldsymbol{I}_3 \in \mathbb{R}^{3\times3}$ is an identity matrix, and $\boldsymbol{O}_3 \in \mathbb{R}^{3\times3}$ is a zero matrix.

The task of turning over is illustrated in Fig. 2.1. As a previous research [36], dynamic constraint is represented by an imaginary shape called friction cone. Contact between the pancake and the spatula is maintained when the resultant force vector of gravity and inertial force fits into the friction cone. This condition is expressed as follows:

$$|\arg(\boldsymbol{g} - \boldsymbol{u}[k]) - \theta[k]| < \arctan\mu, \tag{2.6}$$

where arg is an operator which gives the angle of a given vector, $\boldsymbol{g}$ is a gravitational acceleration vector, and μ is a static friction coefficient. This chapter considers the case of $\mu = 1.0$.

2.2.2 Seq2seq Models

A seq2seq model was implemented as illustrated in Fig. 2.2. Here, for given discrete time indices k_1 and k_2, let $\boldsymbol{x}[k_1 : k_2]$ represent a sequence of vectors $\boldsymbol{x}$ from the k_1-th sample to the k_2-th sample as follows:

$$\boldsymbol{x}[k_1 : k_2] = (\boldsymbol{x}[k_1], \boldsymbol{x}[k_1+1], \ldots, \boldsymbol{x}[k_2]). \tag{2.7}$$

In addition, $\hat{\bullet}$ denotes the samples or sequences of the generated samples from the seq2seq models.

This seq2seq model has the same input/output relationship as implemented in [19]: the encoder receives time series of state vectors,

$$\boldsymbol{x}[C\kappa : C(\kappa+1) - 1], \tag{2.8}$$

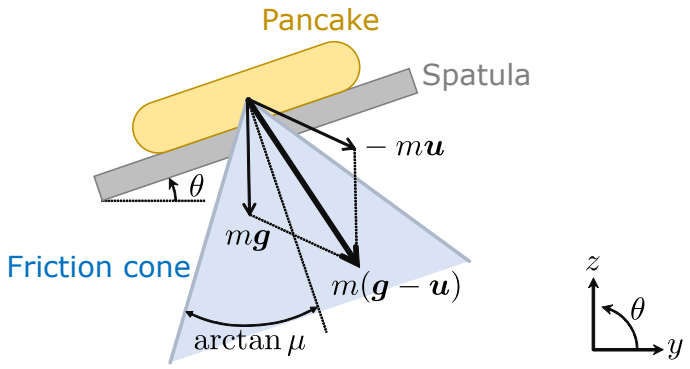


Figure 2.1: Overview of the task of turning a pancake with a spatula. Contact is maintained if the resultant force vector of $m\boldsymbol{g}$ and $-m\boldsymbol{u}$ fits into the friction cone (blue). The apex angle of the friction cone is determined by the static friction coefficient, μ.

where κ and C indicate the discrete time during the encoding phase and the number of samples the seq2seq models process at a time, respectively. The decoder receives deformed state vectors,

$$\hat{\boldsymbol{x}}[Ck+1:C(k+1)], \tag{2.9}$$

and outputs mean vectors and covariant matrices of acceleration input,

$$\hat{\boldsymbol{u}}[C(k+1):C(k+2)-1], \quad \hat{\boldsymbol{S}}_u[C(k+1):C(k+2)-1]. \tag{2.10}$$

Here, k indicates the discrete time during the decoding phase. $\hat{\boldsymbol{u}}[k]$ is a mean vector used as generated acceleration and $\hat{\boldsymbol{S}}_u[k]$ is a covariant matrix used as uncertainty of the outputs [43]. A covariant matrix $\hat{\boldsymbol{S}}_u[k]$ is expressed as follows:

$$\hat{\boldsymbol{S}}_u[k] = \begin{bmatrix} \sigma_y^2[k] & \rho_{yz}[k]\sigma_y[k]\sigma_z[k] & 0 \\ \rho_{yz}[k]\sigma_y[k]\sigma_z[k] & \sigma_z^2[k] & 0 \\ 0 & 0 & \sigma_\theta^2[k] \end{bmatrix}, \tag{2.11}$$

where $\sigma_y^2[k]$, $\sigma_z^2[k]$, and $\sigma_\theta^2[k]$ are variance and $\rho_{yz}[k]$ is a correlation coefficient between y and z. Note that at the first sample in the decoding phase, the decoder receives the following chunk instead of (2.9):

$$(\boldsymbol{x}[0], \boldsymbol{x}[0], \ldots, \boldsymbol{x}[0]). \tag{2.12}$$

By using the mean vectors and the covariant matrices, the state equation (2.1) is rewritten as follows:

$$\hat{\boldsymbol{x}}[k+1] = \boldsymbol{A}\hat{\boldsymbol{x}}[k] + \boldsymbol{B}\hat{\boldsymbol{u}}[k], \tag{2.13}$$

$$\hat{\boldsymbol{S}}_x[k+1] = \boldsymbol{A}\hat{\boldsymbol{S}}_x[k]\boldsymbol{A}^\top + \boldsymbol{B}\hat{\boldsymbol{S}}_u[k]\boldsymbol{B}^\top, \tag{2.14}$$

$$\hat{\boldsymbol{x}}[0] = \boldsymbol{x}[0], \tag{2.15}$$

$$\hat{\boldsymbol{S}}_x[0] = \boldsymbol{O}_6, \tag{2.16}$$

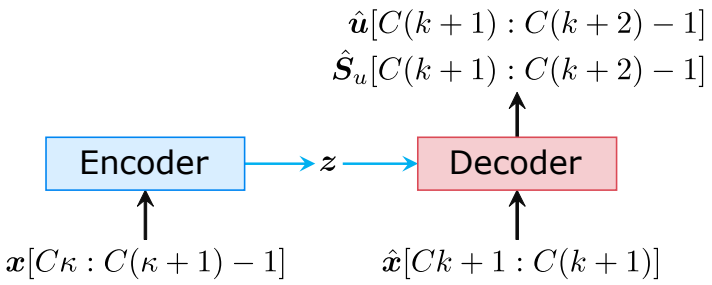


Figure 2.2: Overview of the proposed seq2seq model.

where $\boldsymbol{O}_6 \in \mathbb{R}^{6\times 6}$ is a zero matrix.

The deformation task consists of two objectives: sustaining the original shape and satisfying dynamic constraint. Therefore, a loss function can be expressed as follows:

$$L = \frac{1}{K-1}\sum_{k=0}^{K-2}\left(\alpha_1[k+1]L_1[k+1] + \alpha_2[k]L_2[k]\right), \tag{2.17}$$

where α_1 and α_2 are weight coefficients, L_1 is the degree of sustaining trajectory shapes, L_2 is the degree of satisfying dynamic constraint, and K is the number of samples in a trajectory. L_1 and L_2 can be calculated by inputs and outputs of the seq2seq models. When the loss function minimizes, it is expected that the models are able to convert trajectories to satisfy dynamic constraint while sustaining the original shapes.

2.2.3 Proposed Learning Curriculum

It is difficult to prepare a large dataset of input trajectories and corresponding deformed trajectories. Therefore, unsupervised learning is desired. However, training seq2seq models by unsupervised learning is a challenging task. In supervised learning, the decoder receives the true output samples (*i.e.* teacher data) during training. For example, "Je," "suis," and "étudiant" entered in the decoder in Fig. 1.2 are the teacher data. On the other hand, in unsupervised learning, the decoder cannot receive the true output samples. Instead, the decoder receives samples generated by the decoder itself. Receiving generated samples makes training unstable due to accumulating prediction errors. This issue is remarkable especially in the beginning of training.

To deal with the issue, this chapter proposes a training method based on curriculum learning [44]. Curriculum learning is a training strategy that starts from an easy task and increases the difficulty. Based on the idea, a learning curriculum for learning trajectory deformation under unsupervised learning is proposed here. The proposed learning curriculum increases the amount of deformation gradually as the training progresses. To control the amount of deformation, the degree of dynamic constraint satisfaction is introduced as follows:

$$F_{\mathrm{c}}[k] = \max(0, |\arg(\boldsymbol{g} - \boldsymbol{u}[k]) - \theta[k]| - \arctan\mu), \tag{2.18}$$

which is obtained by (2.6). A trajectory satisfies the dynamic constraint if $F_{\mathrm{c}}[k] = 0$ for

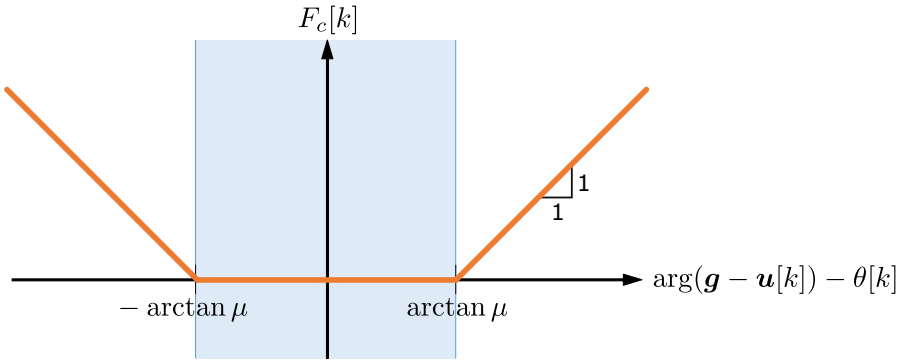


Figure 2.3: Shape of $F_{\mathrm{c}}[k]$, defined in (2.18). The blue area is dynamic constraint. $F_{\mathrm{c}}[k]$ increases as dynamic constraint does not satisfy.

all k. The shape of $F_{\mathrm{c}}[k]$ is illustrated in Fig. 2.3. It is regarded that the amount of deformation is small as $\sup_k F_{\mathrm{c}}[k]$ in a trajectory is small. Moreover, when a trajectory satisfies the dynamic constraint, no deformation is necessary. This case is the easiest deformation task.

Thus, the proposed learning curriculum for trajectory deformation progresses as follows. At first, a seq2seq model as a sequence autoencoder [45] is trained by using trajectories with $\sup_k F_{\mathrm{c}}[k] = 0$. These trajectories satisfy the dynamic constraint (*i.e.*, no deformation is needed). In this time, training process of sequence autoencoder progresses with two steps: teacher forcing and free running. During teacher forcing training, a decoder receives true data. On the other hand, during free running training, the decoder receives samples generated by itself. Scheduled sampling [24] is applied during transition from teacher forcing training to free running training. After a sequence autoencoder is trained, the seq2seq models continue training while increasing $F_{\mathrm{c}}[k]$ gradually. The seq2seq models have to deform given trajectories to satisfy dynamic constraint. In this step, it is expected that the seq2seq models can learn deformation method if the speed of increasing difficulty is small enough. In addition, since the degree of satisfying dynamic constraint has a limit $0 \leq F_{\mathrm{c}}[k] < \pi - \arctan\mu$, finally the seq2seq models have a chance of receiving every possible trajectory.

2.3 Experiments

2.3.1 Implementation of Seq2seq Model

A seq2seq model was implemented according to Section 2.2.2. The implementation is illustrated in Fig. 2.4. The model has two RNNs with two layers in each RNN, with 512 cells in a layer. The encoder and the decoder process $C = 50$ samples at a time.

In Fig. 2.4, the Linear layer calculates as follows:

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \tag{2.19}$$

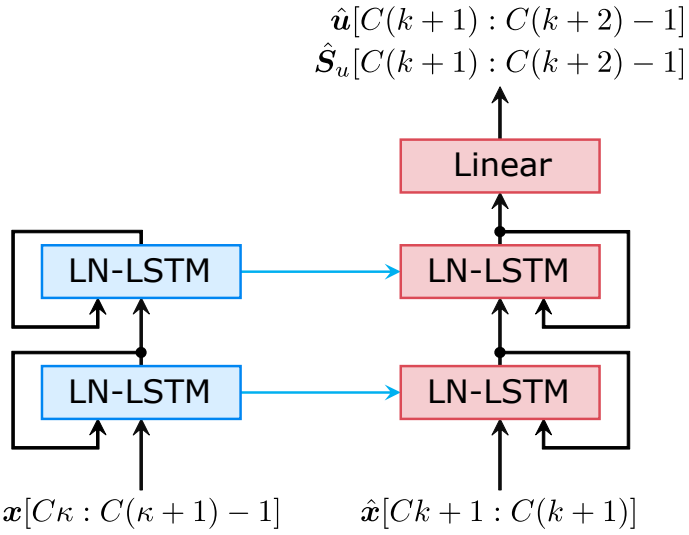


Figure 2.4: Architecture of the proposed seq2seq model.

where $\boldsymbol{x}$ is an input vector, $\boldsymbol{h}$ is an output vector, $\boldsymbol{W}$ is an weight matrix, and $\boldsymbol{b}$ is an bias vector. $\boldsymbol{W}$ and $\boldsymbol{b}$ are adaptive parameters of the layer. The LN-LSTM indicates LSTM with layer normalization [46].

A loss function is defined as (2.17) with the following terms:

$$\alpha_1[k] = \begin{cases} 1 & k < K - 10, \\ 10 & k \geq K - 10, \end{cases} \tag{2.20}$$

$$\alpha_2[k] = 10, \tag{2.21}$$

$$L_1[k] = -\log \Pr\left(\boldsymbol{D}\boldsymbol{x}[k] \,\middle|\, \boldsymbol{D}\boldsymbol{\mu}_x[k], \boldsymbol{D}S_x[k]\boldsymbol{D}^\top\right), \tag{2.22}$$

$$L_2[k] = F_c[k], \tag{2.23}$$

where

$$\Pr(\boldsymbol{x}\,|\boldsymbol{\mu}, \boldsymbol{S}) = \frac{1}{(2\pi)^{\frac{\dim \boldsymbol{x}}{2}}\sqrt{\det S}} \cdot \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top S^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right], \tag{2.24}$$

$$\boldsymbol{D} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \end{bmatrix}. \tag{2.25}$$

The matrix $\boldsymbol{D}$ extracts a position and an attitude from a state vector and normalizes difference of the scale between meters and radians. K is the number of samples in a trajectory. Adam [47] with default parameters was used for optimizing a neural network.

A seq2seq model is trained with 1638400 trajectories, which are generated randomly in simulation. Each trajectory has 100–500 samples. Two kinds of trajectories were used. One is spline curves generated in acceleration space. Hereafter, this chapter calls

them *acceleration type trajectories*. Each acceleration type trajectory has a parameter $\phi_{\max}$, which limits the range of the trajectory as $\sup_k |\arg(\boldsymbol{g} - \boldsymbol{u}[k]) - \theta[k]| < \phi_{\max}$. Acceleration type trajectories are generated by the following algorithm. First, spline curves are generated by three control points $\boldsymbol{a}_0$, $\boldsymbol{a}_1$, and $\boldsymbol{a}_2$, which are sampled from the following sets randomly:

$$\boldsymbol{a}_0 = (0, 0, 0), \tag{2.26}$$

$$\boldsymbol{a}_1, \boldsymbol{a}_2 \in \big\{(a_r, a_\phi, a_\theta)\big| \\ 0 < a_r < 20, |a_\phi| < \phi_{\max}, |a_\theta| < -50\big\}. \tag{2.27}$$

Then, the following calculation is applied to the spline curves:

$$a'_\phi = \phi_{\max} \tanh a_\phi, \tag{2.28}$$

$$a_y = |a'_r| \sin(a'_\phi - \theta), \tag{2.29}$$

$$a_z = |a'_r| \cos(a'_\phi - \theta) - g. \tag{2.30}$$

By setting $\phi_{\max} = \arctan \mu$, acceleration type trajectories are limited as $\sup_k F_{\mathrm{c}}[k] = 0$. These trajectories are used in the proposed learning curriculum. The other one is spline curves generated in position space. Hereafter, this chapter calls them *position type trajectories*. Position type trajectories are generated by three control points, $\boldsymbol{c}_0$, $\boldsymbol{c}_1$, and $\boldsymbol{c}_2$, which are sampled from the following sets randomly:

$$\boldsymbol{c}_0 = (0, 0, 0), \tag{2.31}$$

$$\boldsymbol{c}_1 \in \big\{(y, z, \theta)\big||y| < 0.15, |z| < 0.15, |\theta| < 1.57\big\}, \tag{2.32}$$

$$\boldsymbol{c}_2 \in \big\{(y, z, \theta)\big||y| < 0.3, |z| < 0.3, |\theta| < 3.14\big\}. \tag{2.33}$$

After a position type trajectory is generated, it is transformed to S-curve acceleration/deceleration.

The learning curriculum consists of four steps: 1) training sequence autoencoder by teacher forcing, 2) training sequence autoencoder by free running, 3) training a deformation method with acceleration type trajectories with increasing $\phi_{\max}$, and 4) training a deformation method with position type trajectories. These four steps progress gradually. Training process is composed of units called *epochs*. One epoch consists of training 64 mini-batches and one mini-batch consists of parallel computation of 256 trajectories. This chapter trained a seq2seq model with 100 epochs. In the first 10 epochs, the seq2seq model is trained as sequence autoencoder with teacher forcing. The input trajectories are acceleration type trajectories with $\phi_{\max} = \arctan \mu = \pi/4$ so that $F_{\mathrm{c}} = 0$. Since these trajectories satisfy dynamic constraint, no deformation is needed. In the next 10 epochs, teacher forcing and free running are mixed by using scheduled sampling. Probability to use teacher forcing decreases from 1 to 0 linearly. Then, the sequence autoencoder is trained with free running in the next 10 epochs. After that, $\phi_{\max}$ increases to π over the next 10 epochs. The seq2seq model continues dealing with slightly larger deformation than before. Then, the training continues with $\phi_{\max} = \pi$ in the next 10 epochs. After the 60th epoch, the type of input trajectories changes to position type trajectories gradually.

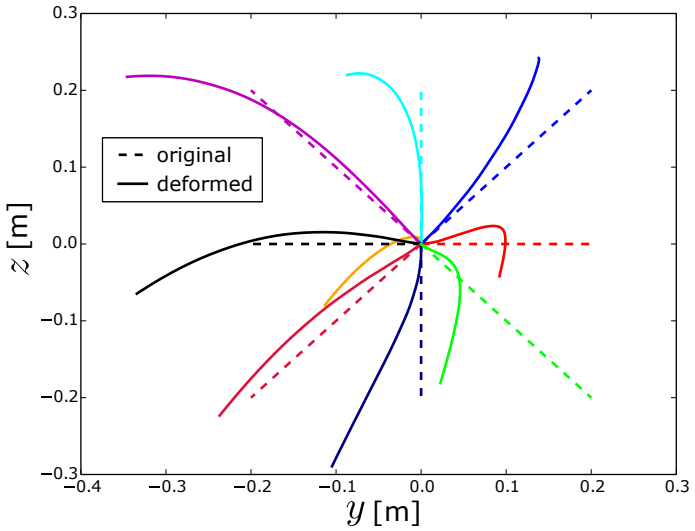


Figure 2.5: Results of deformation. The dashed lines indicate original trajectories and the solid lines indicate deformed trajectories by the seq2seq model.

One of two types of trajectories are selected randomly and the probability of acceleration type trajectories decreases 1 to 0 over 20 epochs. Finally, the seq2seq model is trained with position type trajectories in the last 10 epochs.

2.3.2 Simulation Results

A seq2seq model as described above was implemented. First, the ability to deform trajectories of the seq2seq model was confirmed.

Fig. 2.5 shows the results of deformation. This figure shows nine trajectories from the origin to points $y = 0, \pm 2.0$, $z = 0, \pm 2.0$, $\theta = 2.5$. The trajectories are deformed while sustaining the original direction of motion. In addition, according to Fig. 2.6, the deformed trajectory satisfies the dynamic constraint. The other trajectories are also deformed to satisfy dynamic constraint. Fig. 2.7 shows the amount of deformation. In the first half of the trajectory, the deformation amount increased as the trajectory progress. However, the deformation amount of angle decreased in the end of the trajectory. Also, the deformation amount of position became gentle. It seems to be thanks to the weighting in (2.20). Thus, even if a trajectory needs deformation, balances of the deformation amount in the trajectory can be changed.

2.3.3 Experimental Result

An experiment of the trained seq2seq model with a manipulator was conducted. A six degrees of freedom (DOF) manipulator "MOTOMAN-MH3F," supplied by Yaskawa Electric, was used. The manipulator has a gripping mechanism on the tip of the arm. A spatula is fixed on the gripping mechanism.

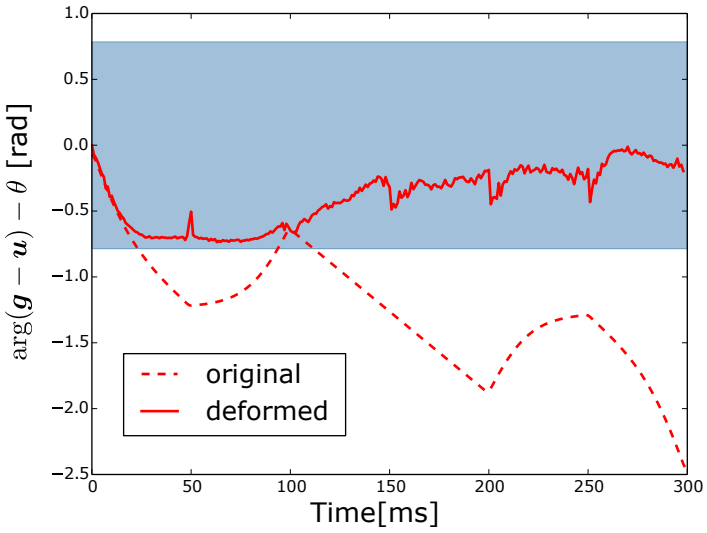


Figure 2.6: Result of $\arg(\boldsymbol{g}-\boldsymbol{u}[k])-\theta[k]$ of the trajectory from the origin $(y, z, \theta) = (0, 0, 0)$ to the point $(y, z, \theta) = (0.2, 0, 2.5)$ (red colored trajectory in Fig. 2.5). The blue area satisfies the dynamic constraint. The deformed trajectory is within the area.

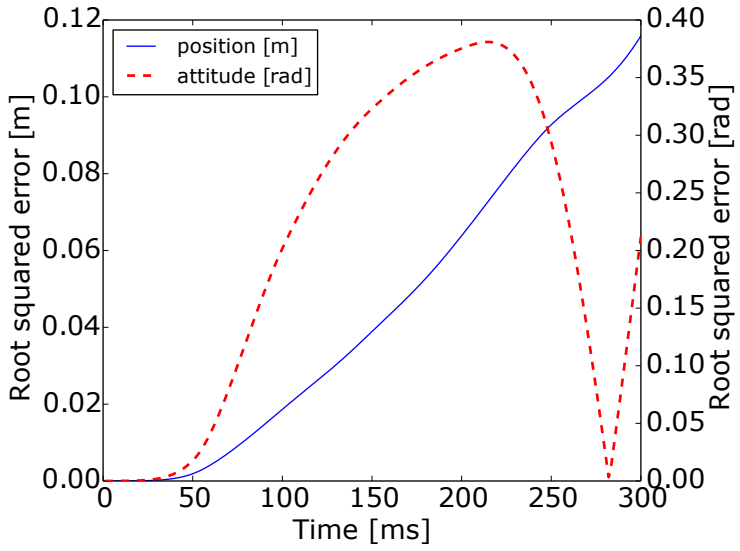


Figure 2.7: Amount of deformation of the trajectory from the origin $(y, z, \theta) = (0, 0, 0)$ to the point $(y, z, \theta) = (0.2, 0, 2.5)$ (red colored trajectory in Fig. 2.5).

A P-D controller with a disturbance observer (DOB) [48, 49] was implemented. The proportional position gain is 350, the proportional attitude gain is 600, the derivative position gain is 60, and the derivative attitude gain is 90. The cutoff frequency of the DOB is set to 12.57 [rad/sec]. A trajectory of turning motion, which does not satisfy the dynamic constraint (*i.e.*, fails the task), was prepared. The trajectory is deformed by the seq2seq model and provided to the control system as a reference trajectory. The original

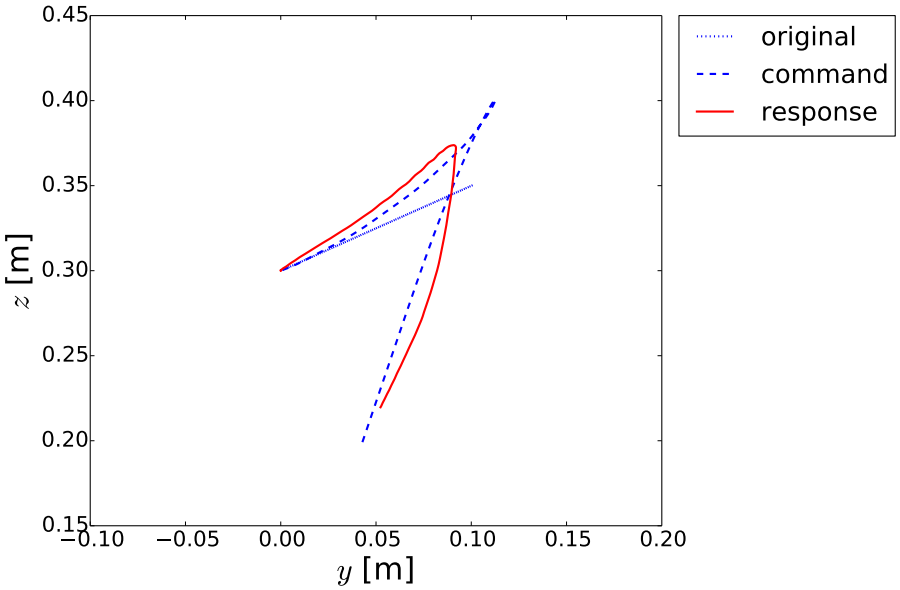


Figure 2.8: The original trajectory (blue dotted line), the deformed trajectory (blue dashed line), and the control response trajectory (red solid line).

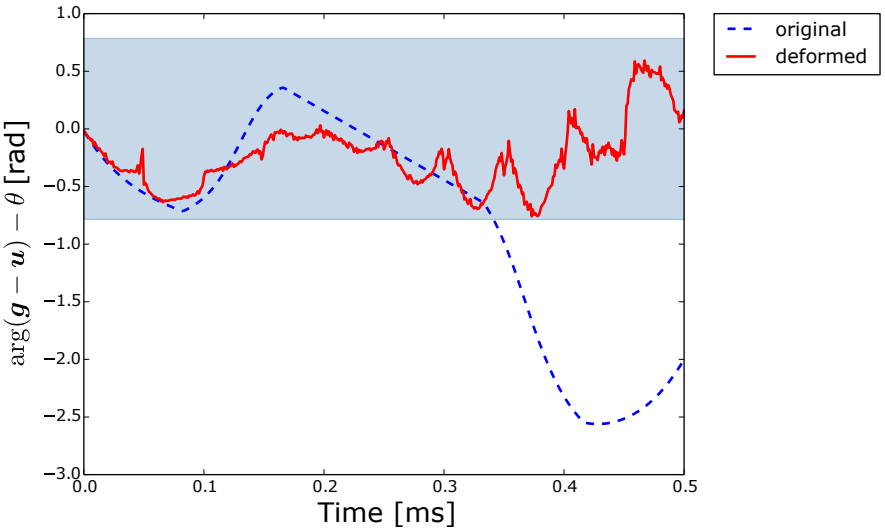


Figure 2.9: Result of $\arg(\boldsymbol{g} - \boldsymbol{u}[k]) - \theta[k]$ of the trajectory in Fig. 2.8. The blue area satisfies the dynamic constraint. The deformed trajectory is within the area.

trajectory, the deformed trajectory, and the control response are shown in Fig. 2.8. The original trajectory is deformed to satisfy dynamic constraint as shown in Fig. 2.9. The control response has deviations. However, seeing snapshots of the experiment shown in Fig. 2.10, the pancake has sustained contact with the spatula. It is observed that the manipulator succeeded the task of turning over a pancake.

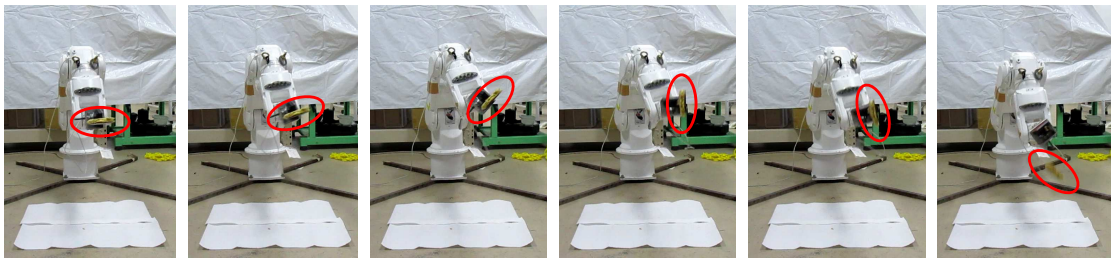


Figure 2.10: Snapshots of the turning over motion.

2.4 Conclusion

It is difficult to obtain a repertory of trajectories that has enough diversity as well as satisfying the dynamic constraints. For this issue, trajectory deformation method is an effective approach. This chapter proposed a training method for seq2seq models to deform given trajectories to satisfy dynamic constraint. In addition, a learning curriculum which allows the seq2seq model to learn a method of trajectory deformation under unsupervised learning was proposed. The curriculum is designed according to a degree to which a trajectory satisfies dynamic constraint. Users can obtain trajectories for dynamic manipulation by inputting outlines of motion to the trained seq2seq model.

This chapter examined a task of turning over pancakes with a spatula as an example. The validity of the proposed method was confirmed by simulation and an experiment.

Chapter 3

Training of Seq2seq Models under Discontinuous Dynamics[1]

In the previous chapter, a training method for seq2seq models to learn diverse trajectories has been proposed. However, the environmental model was a simple linear model. In more complicated environmental models, training methods using backpropagation often fail due to gradient-vanishing zones.

In this chapter, therefore, seq2seq learning with discontinuous environmental models is explained. A training curriculum is proposed.

3.1 Introduction

Among the many tasks of nonprehensile manipulation, manipulating objects along the desired trajectories is difficult. Such tasks have more strict requirements in both kinematics and dynamics than when only the goals are specified. To handle such complex issues, a seq2seq model-based method was proposed in the previous chapter. This method, however, considered the dynamic constraint only as a penalty in the objective function. The seq2seq models only handled the case where the object is fixed to the robot. Therefore, the seq2seq models should be extended to cases where complex dynamics exist, such as the contact models between the robots and the objects.

This chapter discusses the sliding manipulation [50–52], which is a typical example including the contact models between objects and tools such as frying pans fixed onto the robots. To handle this task, the seq2seq models should consider two different factors: static friction and dynamic friction. Particularly, static friction operates as if the object is fixed onto the frying pan. Hence, training losses vanish during backpropagation [23].

This chapter aims to handle the issue by applying curriculum learning [44]. This chapter proposes a training curriculum that does not use the contact model at the beginning

[1]The contents in this chapter were published in: K. Kutsuzawa, S. Sakaino, and T. Tsuji, "Sequence-to-sequence model for trajectory planning of nonprehensile manipulation including contact model," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3606–3613, 2018.

of the training. The contributions of this chapter are as follows:

1. This chapter proposes seq2seq models for motion planning considering nonlinear contact models. This chapter uses the sliding manipulation as an example. The proposed seq2seq models are trained to manipulate objects on frying pans along the given trajectories by shaking.

2. This chapter proposes a training curriculum to handle nonlinear contact models. The proposed curriculum starts training without nonlinear contact models. It brings the seq2seq models outside the gradient-vanishing zone that is caused by static friction.

3. This chapter shows that the seq2seq models trained by the proposed curriculum can handle fluctuations of friction parameters even without special training.

The remainder of this chapter consists of the following sections. Section 3.2 explains the proposed neural model and the physics model. Section 3.3 explains the proposed training method based on curriculum learning. Section 3.4 describes the concrete implementation of the proposed method. Section 3.5 shows the validity of the proposed method by simulation. Finally, Section 3.6 presents the conclusion and future works.

3.2 Sequence-to-Sequence Model for Sliding Manipulation

3.2.1 Physics Model

This study uses two-dimensional physics and is a typical example that includes nonlinear contact models. Fig. 3.1 shows an overview of the physics model. An object is placed on a frying pan. Gravity is exerted to the normal direction to the frying pan. The task is to manipulate the object along a desired trajectory by moving the frying pan.

Let $\boldsymbol{p}_{\mathrm{p}}[k]$ and $\boldsymbol{p}_{\mathrm{o}}[k]$ denote the position of the frying pan and the object, at the k-th sample, respectively. In addition, let

$$\boldsymbol{p}_{\mathrm{d}}[k] \equiv \boldsymbol{p}_{\mathrm{o}}[k] - \boldsymbol{p}_{\mathrm{p}}[k] \tag{3.1}$$

denote the relative position of the object with respect to the frying pan. Let $\boldsymbol{x}_{\mathrm{p}}[k]$ denote a state vector of the frying pan that comprises the position $\boldsymbol{p}_{\mathrm{p}}[k]$ and the velocity $\dot{\boldsymbol{p}}_{\mathrm{p}}[k]$ at the k-th sample, as follows:

$$\boldsymbol{x}_{\mathrm{p}}[k] \equiv \left[\boldsymbol{p}_{\mathrm{p}}^{\top}[k], \dot{\boldsymbol{p}}_{\mathrm{p}}^{\top}[k]\right]^{\top} = \left[x_{\mathrm{p}}[k], y_{\mathrm{p}}[k], v_{x_{\mathrm{p}}}[k], v_{y_{\mathrm{p}}}[k]\right]^{\top}. \tag{3.2}$$

Similarly, $\boldsymbol{x}_{\mathrm{o}}[k]$ and $\boldsymbol{x}_{\mathrm{d}}[k]$ are also defined.

The state equations of the frying pan and the object are expressed as follows:

$$\boldsymbol{x}_{\mathrm{p}}[k+1] = \boldsymbol{A}\boldsymbol{x}_{\mathrm{p}}[k] + \boldsymbol{B}\boldsymbol{u}_{\mathrm{p}}[k], \tag{3.3}$$

$$\boldsymbol{x}_{\mathrm{d}}[k+1] = \boldsymbol{A}\boldsymbol{x}_{\mathrm{d}}[k] + \boldsymbol{B}\boldsymbol{u}_{\mathrm{d}}[k], \tag{3.4}$$

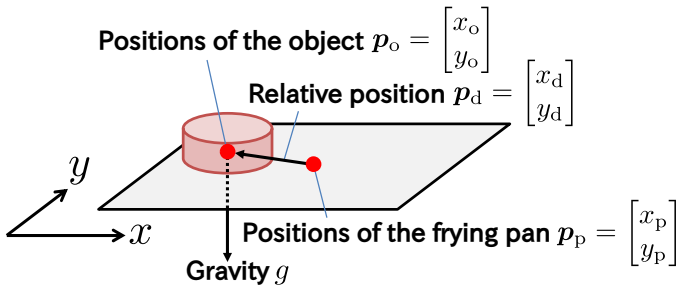


Figure 3.1: Overview of the physics model.

where,

$$\boldsymbol{u}_{\mathrm{p}}[k] = \left[a_{x_{\mathrm{p}}}[k], a_{y_{\mathrm{p}}}[k]\right]^{\top}, \tag{3.5}$$

$$\boldsymbol{u}_{\mathrm{d}}[k] = \left[a_{x_{\mathrm{d}}}[k], a_{y_{\mathrm{d}}}[k]\right]^{\top}, \tag{3.6}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_2 & \Delta t \boldsymbol{I}_2 \\ \boldsymbol{O}_2 & \boldsymbol{I}_2 \end{bmatrix}, \tag{3.7}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{\Delta t^2}{2} \boldsymbol{I}_2 \\ \Delta t \boldsymbol{I}_2 \end{bmatrix}. \tag{3.8}$$

$\boldsymbol{u}_{\mathrm{p}}[k]$ and $\boldsymbol{u}_{\mathrm{d}}[k]$ are the acceleration inputs at the k-th sample and Δt is the sampling interval. $\boldsymbol{I}_2 \in \mathbb{R}^{2\times 2}$ is an identity matrix and $\boldsymbol{O}_2 \in \mathbb{R}^{2\times 2}$ is a zero matrix. These state equations are used to obtain the next state vectors, $\boldsymbol{x}_{\mathrm{p}}[k+1]$ and $\boldsymbol{x}_{\mathrm{d}}[k+1]$, from the given acceleration inputs, $\boldsymbol{u}_{\mathrm{p}}[k]$ and $\boldsymbol{u}_{\mathrm{d}}[k]$, respectively. The acceleration inputs $\boldsymbol{u}$ correspond to the force applied to the rigid objects.

The contact model between the frying pan and the object is implemented as the following Coulomb friction model according to [52]:

$$\boldsymbol{u}_{\mathrm{d}}[k] = \begin{cases} -\boldsymbol{u}_{\mathrm{p}}[k] - \mu' g \frac{\dot{\boldsymbol{p}}_{\mathrm{d}}[k]}{\|\dot{\boldsymbol{p}}_{\mathrm{d}}[k]\|} & \text{if } \dot{\boldsymbol{p}}_{\mathrm{d}}[k] \neq \boldsymbol{0} \\ -\boldsymbol{u}_{\mathrm{p}}[k] + \mu g \frac{\boldsymbol{u}_{\mathrm{p}}[k]}{\|\boldsymbol{u}_{\mathrm{p}}[k]\|} & \text{if } \dot{\boldsymbol{p}}_{\mathrm{d}}[k] = \boldsymbol{0} \text{ and } \|\boldsymbol{u}_{\mathrm{p}}[k]\| \geq \mu g \\ \boldsymbol{0} & \text{if } \dot{\boldsymbol{p}}_{\mathrm{d}} = \boldsymbol{0} \text{ and } \|\boldsymbol{u}_{\mathrm{p}}[k]\| < \mu g \end{cases}. \tag{3.9}$$

Here, $g = 9.8$ m/s^2 is the gravitational acceleration. μ and μ' indicate the static friction coefficient and the dynamic friction coefficient, respectively. The equation above related to acceleration was derived by dividing the equation of force by the mass. Because $\boldsymbol{u}_{\mathrm{d}}$ is the acceleration of the object with respect to the frying pan, $\boldsymbol{u}_{\mathrm{d}}$ is affected by the friction force and the inertial force of $\boldsymbol{u}_{\mathrm{p}}$. The two forces above are the entire force applied to the object.

3.2.2 Seq2seq Model Architecture

The seq2seq model is used as the motion-to-motion conversion, as proposed in Chapter 2. Our model has the following characteristics compared to the basic seq2seq model.

- The encoder receives the command trajectories of the object. Subsequently, the decoder generates the acceleration reference to the robot to manipulate the object along the command trajectories.
- Our model processes multiple state vectors at a time to reduce the number of RNN iterations without downsampling [19]. This method can reduce the memory consumption of computers and enable stable training.

Fig. 3.2 illustrates the primary architecture of the proposed seq2seq model.

In the encoding phase, the encoder of a seq2seq model receives the time-series of state vectors of the object's command trajectories. In each iteration, the encoder receives part of the time-series as follows:

$$\begin{aligned}&\boldsymbol{x}_{\mathrm{d}}[C\kappa : C(\kappa+1)-1]\\&\quad=(\boldsymbol{x}_{\mathrm{d}}[C\kappa],\ldots,\boldsymbol{x}_{\mathrm{d}}[C(\kappa+1)-1])\,.\end{aligned}\tag{3.10}$$

Here, κ is the index of iterations of the encoding. C is the number of the samples the seq2seq models process at a time [19]. That is, the encoder receives C samples in each iteration. After the encoder receives the whole command trajectories, it copies its internal representation to the decoder's internal memory. Subsequently, the decoding phase commences.

In the decoding phase, the decoder receives/generates C samples at a time. An overview of the decoding process in an iteration is illustrated in Fig. 3.3. In each iteration, the decoder receives part of the time-series of state vectors as follows:

$$\hat{\boldsymbol{x}}_{\mathrm{p}}[Ck+1 : C(k+1)],\ \hat{\boldsymbol{x}}_{\mathrm{d}}[Ck+1 : C(k+1)].\tag{3.11}$$

Here, k is the index of iterations of the decoding. Subsequently, the decoder generates the acceleration references of the frying pan,

$$\hat{\boldsymbol{u}}_{\mathrm{p}}[C(k+1) : C(k+2)-1],\tag{3.12}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{u}_{\mathrm{p}}}[c(k+1) : c(k+2)-1].\tag{3.13}$$

Here, $\hat{\boldsymbol{S}}_{\boldsymbol{u}_{\mathrm{p}}}[k] \in \mathbb{R}^{2\times 2}$ is a covariance matrix of the acceleration $\boldsymbol{u}_{\mathrm{p}}$. They are used only for training [43]. By inputting the acceleration to the robot, the next C samples of (3.11) is obtained as follows:

$$\hat{\boldsymbol{x}}_{\mathrm{p}}[C(k+1)+1 : C(k+2)],\tag{3.14}$$

$$\hat{\boldsymbol{x}}_{\mathrm{d}}[C(k+1)+1 : C(k+2)],\tag{3.15}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{p}}}[C(k+1)+1 : C(k+2)],\tag{3.16}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{d}}}[C(k+1)+1 : C(k+2)].\tag{3.17}$$

Here, $\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{p}}}$ and $\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{d}}}$ are calculated based on the state equations (3.3) and (3.4) as follows:

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{p}}}[k+1] = \boldsymbol{A}\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{p}}}[k]\boldsymbol{A}^{\top} + \boldsymbol{B}\hat{\boldsymbol{S}}_{\boldsymbol{u}_{\mathrm{p}}}[k]\boldsymbol{B}^{\top},\tag{3.18}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{d}}}[k+1] = \boldsymbol{A}\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{d}}}[k]\boldsymbol{A}^{\top} + \boldsymbol{B}\hat{\boldsymbol{S}}_{\boldsymbol{u}_{\mathrm{d}}}[k]\boldsymbol{B}^{\top}.\tag{3.19}$$

$\hat{\boldsymbol{x}}_{\mathrm{p}}$ and $\hat{\boldsymbol{x}}_{\mathrm{d}}$ are fed back to the decoder at the next iteration.

3.3 Training Method

3.3.1 Objective Function

The seq2seq model is expected to manipulate the object along the given command trajectories. Therefore, the objective of the training is to minimize the errors between the command trajectories and the reproduced trajectories of the object by the seq2seq models.

The objective of the training is to bring the generated trajectories of the relative positions of the object $\hat{\boldsymbol{p}}_{\mathrm{d}}[0:K-1]$ closer to the command trajectories $\boldsymbol{p}_{\mathrm{d}}[0:K-1]$. Such input–output relationship can be obtained by minimizing the loss function L defined as follows:

$$L = \frac{1}{K-1}\sum_{k=1}^{K-1}(\alpha_1 L_1[k] + \alpha_2 L_2[k] + \alpha_3 L_3[k]), \tag{3.20}$$

where

$$\begin{aligned} L_1[k] =& \log\left(2\pi\sqrt{\det\hat{\boldsymbol{S}}_{\boldsymbol{p}_{\mathrm{d}}}[k]}\right) \\ &+ (\boldsymbol{p}_{\mathrm{d}}[k] - \hat{\boldsymbol{p}}_{\mathrm{d}}[k])^{\top}\hat{\boldsymbol{S}}_{\boldsymbol{p}_{\mathrm{d}}}^{-1}[k](\boldsymbol{p}_{\mathrm{d}}[k] - \hat{\boldsymbol{p}}_{\mathrm{d}}[k]), \end{aligned} \tag{3.21}$$

$$L_2[k] = \max(0, \|\hat{\boldsymbol{p}}_{\mathrm{p}}[k]\| - p_{\mathrm{lim}}), \tag{3.22}$$

$$L_3[k] = \|\hat{\boldsymbol{u}}_{\mathrm{p}}[k] - \hat{\boldsymbol{u}}_{\mathrm{p}}[k-1]\|. \tag{3.23}$$

Here, α_1, α_2, and α_3 are the weight coefficients at the k-th sample. $L_1[k]$ is a negative log-likelihood of a Gaussian distribution for a given $\hat{\boldsymbol{p}}_{\mathrm{d}}[k]$. $L_2[k]$ is a penalty for the range of motion of the frying pan. It increases when p_{p} is larger than the limit p_{lim}. $L_3[k]$ is a penalty for the change in $\boldsymbol{u}_{\mathrm{p}}[k]$, which corresponds to the jerk. In addition, $\hat{\boldsymbol{S}}_{\boldsymbol{p}_{\mathrm{d}}}[k] \in \mathbb{R}^{2\times 2}$ refers to the covariance matrix of the position $\hat{\boldsymbol{p}}_{\mathrm{d}}[k]$, *i.e.*, the upper left 2×2 part of $\hat{\boldsymbol{S}}_{\boldsymbol{x}_{\mathrm{d}}}[k]$. p_{lim} refers to the limit of the movable range of the frying pan. Its value is determined by considering the arm lengths of the manipulators. All variables in L can be calculated using only the inputs and outputs of the seq2seq model. Other external teaching signals are not necessary. Therefore, it is not necessary to prepare the trajectories of the frying pan $\boldsymbol{x}_{\mathrm{p}}$, which realizes the command trajectories $\boldsymbol{x}_{\mathrm{d}}$. Hence, arbitrary command trajectories such as spline curves generated automatically can be used regardless of the contact model.

3.3.2 Training Strategy Based on Curriculum Learning

During the training of the seq2seq models with (3.20) by backpropagation, the gradient of the training loss L back-propagates through the contact model in (3.9). This gradient, however, loses at the contact model when $\dot{\boldsymbol{p}}_{\mathrm{d}} = \boldsymbol{0}$. Subsequently, the connection weights in the seq2seq models will no longer be updated by the gradient descent methods. Therefore, the training may fail because of the contact model.

To handle such problem, training should progress outside the gradient-vanishing zone. Curriculum learning [44] is applied herein. In the beginning of the training, a simple task is employed: reproducing the positions of the frying pan. Trajectories, $\boldsymbol{x}_{\mathrm{p}}[0:K-1]$ and

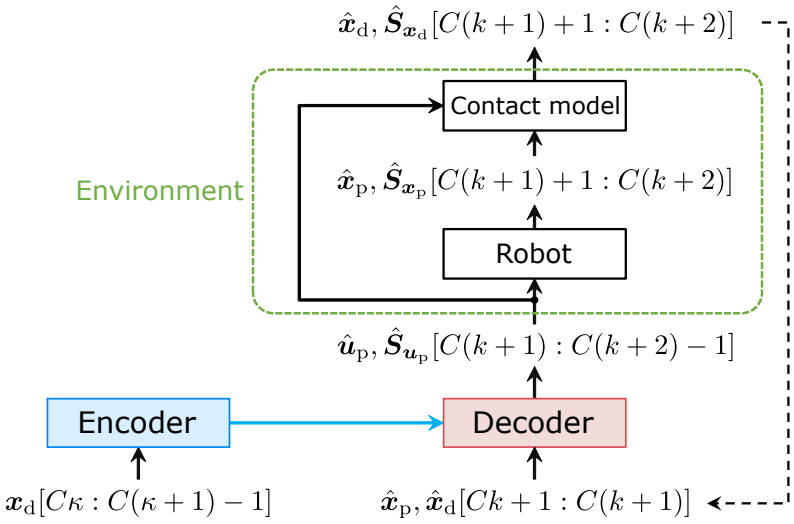


Figure 3.2: Architecture of the proposed seq2seq model.

$\boldsymbol{x}_\mathrm{d}[0:K-1]$, are generated with a contact model as described in Section 3.2.1. The following loss function, L', is used:

$$L' = \frac{1}{K-1}\sum_{k=1}^{K-1}\Bigg[\log\left(2\pi\sqrt{\det\hat{\boldsymbol{S}}_{\boldsymbol{p}_\mathrm{p}}[k]}\right) + (\boldsymbol{p}_\mathrm{p}[k]-\hat{\boldsymbol{p}}_\mathrm{p}[k])^\top\hat{\boldsymbol{S}}_{\boldsymbol{p}_\mathrm{p}}^{-1}[k](\boldsymbol{p}_\mathrm{p}[k]-\hat{\boldsymbol{p}}_\mathrm{p}[k])\Bigg]. \tag{3.24}$$

Here, $\hat{\boldsymbol{S}}_{\boldsymbol{p}_\mathrm{p}}[k] \in \mathbb{R}^{2\times 2}$ refers to the covariance matrix of the position $\hat{\boldsymbol{p}}_\mathrm{p}[k]$, *i.e.*, the upper left 2×2 part of $\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{p}}[k]$. This loss function resembles L_1 in (3.21) and refers to the positions of the frying pan instead of the object. During the training with this task, the training losses L' do not back-propagate through the contact model but to the seq2seq models directly.

After bringing the seq2seq models outside the gradient-vanishing zone, the contact model can be used for training. By switching to such end-to-end training, it is not required to specify the motion of the frying pan. Therefore, the additional penalties can be used in the loss functions as $L_2[k]$ and $L_3[k]$ in (3.20).

3.4 Implementation

3.4.1 Implementation of Seq2seq Model

Here, an implementation of the seq2seq model is described. Fig. 3.4 shows its architecture. The encoder comprises two recurrent layers with 1024 units. The decoder comprises two recurrent layers with 1024 units and a feed-forward layer (indicated as Linear in Fig. 3.4) with 500 units. Each unit in the recurrent layers is implemented as LSTMs with tanh activation functions. The seq2seq model receives/generates $C = 100$

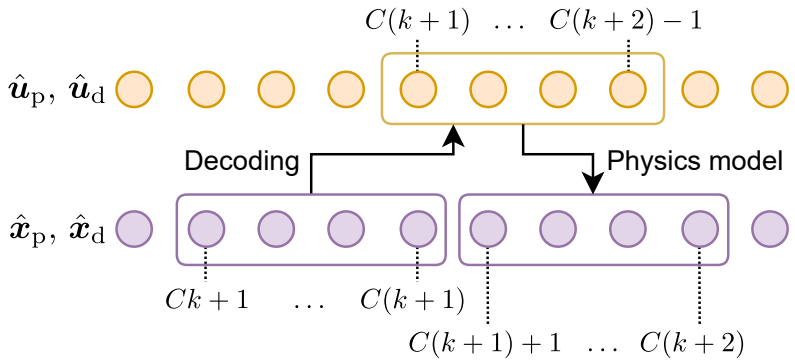


Figure 3.3: Relationship between time series of $\hat{\boldsymbol{u}}_\mathrm{p}$, $\hat{\boldsymbol{x}}_\mathrm{p}$, $\hat{\boldsymbol{u}}_\mathrm{d}$, and $\hat{\boldsymbol{x}}_\mathrm{d}$ in the decoding phase. Each circle indicates a variable at a sample. Here illustrates the case of $C = 4$ as an example.

samples at a time to reduce the number of RNN iterations [19]. This method can reduce the computational costs for backpropagation through time.

3.4.2 Implementation of Training

The training data are generated by simulation. Two types of command trajectories are used: one is generated by the physics model and the other one is generated from B-spline curves. The former one is generated using the physics model explained in Section 3.2.1. By generating the shaking motion of the frying pan $\boldsymbol{u}_\mathrm{p}[0 : K-2]$ and inputting them to the contact model, the trajectories of the frying pan $\boldsymbol{x}_\mathrm{p}[0 : K-1]$ and the object $\boldsymbol{x}_\mathrm{d}[0 : K-1]$ are obtained. $\boldsymbol{x}_\mathrm{p}[0 : K-1]$ is used when the seq2seq models are trained with L'. The latter one is generated from the B-spline curves of the acceleration of the object $\boldsymbol{u}_\mathrm{d}[0 : K-2]$ regardless the contact models. By inputting these acceleration curves to the state equations, the trajectories of the object $\boldsymbol{x}_\mathrm{d}[0 : K-1]$ are obtained. In both types of command trajectories, the length are set to between $500 \leq K \leq 1500$. The sampling interval is set to $\Delta t = 1$ ms. In addition, the friction parameters are set to $\mu' = 0.5$ and $\mu = 1.0$.

The training comprises 50 epochs, where each epoch consists of the training of 8192 trajectories. A training curriculum progresses as described in Table 3.1. In the loss function L in (3.20), $\alpha_1 = 1$, $\alpha_2 = 10$, $\alpha_3 = 1$, and $p_\mathrm{lim} = 0.2$ m were used, respectively. The ratio among α_1, α_2, and α_3 was determined by the approximate amounts of gradients

of each term as follows:

$$\frac{\partial L_1[k]}{\partial \boldsymbol{u}_{\mathrm{p}}[k-1]} \approx 2(\boldsymbol{p}_{\mathrm{d}}[k] - \hat{\boldsymbol{p}}_{\mathrm{d}}[k])^{\top} \hat{\boldsymbol{S}}_{p_{\mathrm{d}}}^{-1}[k] \Delta t^2$$
$$\approx 10^{-3} \cdot (10^{-4})^{-1} \cdot (10^{-3})^2 = 10^{-5}, \tag{3.25}$$
$$\frac{\partial L_2[k]}{\partial \boldsymbol{u}_{\mathrm{p}}[k-1]} \approx \Delta t^2 \approx 10^{-6}, \tag{3.26}$$
$$\frac{\partial L_3[k]}{\partial \boldsymbol{u}_{\mathrm{p}}[k-1]} \approx 1, \tag{3.27}$$

where it is assumed that the positional relative error follows a Gaussian distribution with $\mu = 1$ mm and $\sigma = 1$ cm. The weights were selected such that the gradients of each term are equal and subsequently adjusted experimentally. Based on the approximate gradients, $\alpha_1 = 1$, $\alpha_2 = 10$, and $\alpha_3 = 10^{-5}$ are preferred. However, α_3 was set to a large value as described above to make the jerk small such that the manipulators can follow the deformed trajectories. Adam [47] with default settings was used for optimizing the seq2seq models.

Table 3.1: Training Curriculum

Stage	epoch e	Loss function	Command trajectories.
1	1–10	L'	Generated by the two-dimensional physics model.
2	11–20	L' with probability $\frac{e-10}{10}$ and L with probability $1 - \frac{e-10}{10}$.	Generated by the two-dimensional physics model.
3	21–30	L	Generated by the two-dimensional physics model.
4	31–40	L	Generated by the two-dimensional physics model with probability $\frac{e-30}{10}$ and by B-spline curves with probability $1 - \frac{e-30}{10}$.
5	41–50	L	Generated by B-spline curves.

3.5 Simulation

3.5.1 Implementation of Simulation

Two simulation environments were used: the two-dimensional physics model as described in Section 3.2.1 and a manipulator in pybullet.

Fig. 3.5 shows the setup of the simulation in pybullet. A six DOF manipulator was implemented. A frying pan is fixed at the tip of the manipulator. In addition, a cylinder-shaped object stands on the frying pan with $\mu = \mu' = 0.5$. The frying pan is controlled by a PD controller as illustrated in Fig. 3.6. This controller controls the position and attitude of the frying pan: x, y, z, roll, pitch, and yaw. A DOB is applied to the manipulator. $\boldsymbol{\theta}^{\mathrm{res}}$ are the joint angles and $\boldsymbol{\tau}^{\mathrm{ref}}$ are the joint torque references. $\hat{\boldsymbol{\tau}}^{\mathrm{dis}}$ indicates the estimated

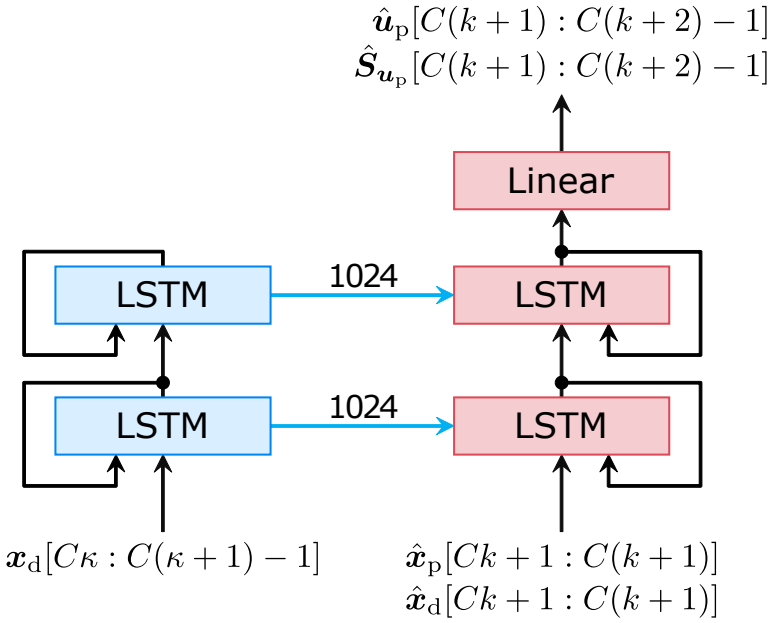


Figure 3.4: Implementation of the proposed seq2seq model.

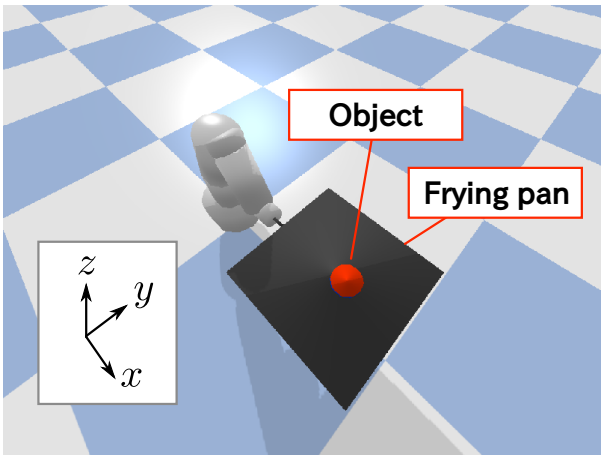


Figure 3.5: Overview of the pybullet simulation.

disturbance torque by the DOB. The commands to the PD controller at the k-th sample, $\boldsymbol{q}_{\mathrm{p}}^{\mathrm{cmd}}[k]$, is defined as follows:

$$\boldsymbol{q}_{\mathrm{p}}^{\mathrm{cmd}}[k] = \left[\hat{x}_{\mathrm{p}}[k] + 0.65, \hat{y}_{\mathrm{p}}[k], 0.3, 0.0, \frac{\pi}{2}, 0.0\right]^{\top} \tag{3.28}$$

where $\hat{x}_{\mathrm{p}}[k]$ and $\hat{y}_{\mathrm{p}}[k]$ are calculated by (3.3) with an output of the seq2seq model, $\hat{\boldsymbol{u}}_{\mathrm{p}}[k-1]$. In addition, $\boldsymbol{K}_{\mathrm{p}} = \mathrm{diag}[800, 800, 800, 1000, 1000, 1000]$ and $\boldsymbol{K}_{\mathrm{d}} = \mathrm{diag}[100, 100, 100, 150, 150, 150]$ are control gains. $\boldsymbol{J}$ is a Jacobian matrix and $\boldsymbol{M}$ is an inertial matrix of the manipulator.

3.5.2 Results

Three seq2seq models were trained as follows:

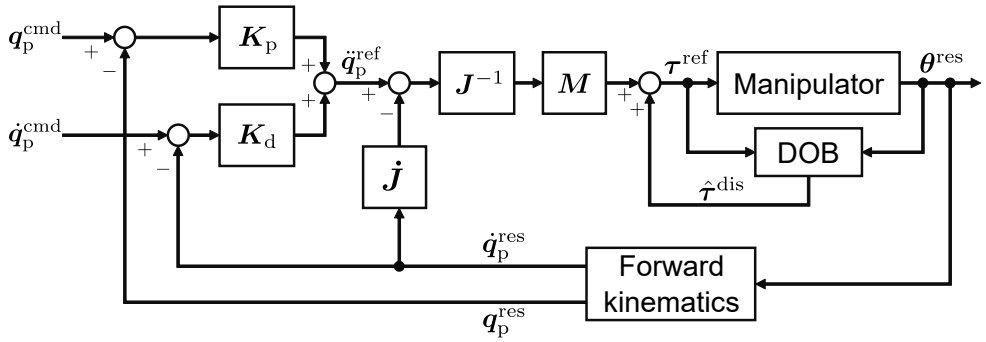


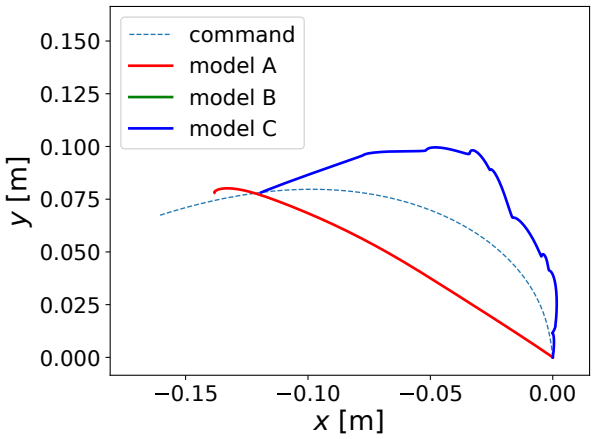
Figure 3.6: Control system for the manipulator in pybullet.

1. A seq2seq model trained by the proposed curriculum in Table 3.1 (model A).
2. A seq2seq model trained without curriculum learning: with only stage 3 in Table 3.1 for 50 epochs (model B).
3. A seq2seq model trained with only L': with only stage 1 in Table 3.1 for 50 epochs (model C).

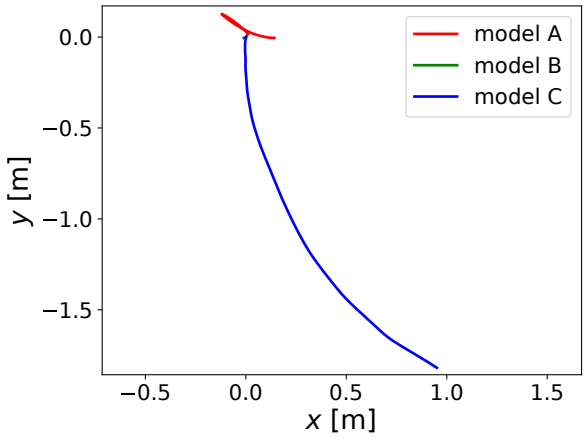
Model A is which the proposed curriculum was applied to. The other two models, model B and model C, were prepared for comparison. The number of training trajectories is the same for each model.

Initially, these models were evaluated in an ideal situation. The physics model as explained in Section 3.2.1 was used. It is the same physics model used in the training. Fig. 3.7 shows the trajectories of the objects and frying pans. All trajectories start from the origin, that is, $\hat{\boldsymbol{p}}_{\mathrm{p}}[0] = \hat{\boldsymbol{p}}_{\mathrm{d}}[0] = [0, 0]^\top$. The length of the command trajectory was set to $K = 1200$. It took 1.93 s to generate each trajectory with Intel Core i7-6700, that is, the models generated 622 samples/s. Model B did not move both the frying pan and the object, whereas model C could manipulate the object. The primary difference between the two models is whether the training loss back-propagates through the contact model. Therefore, the result implies that the contact model interfered with the training. Model C, however, moved the frying pan over one meter. Such large motions cannot be reproduced by most manipulators. Meanwhile, model A moved the object within a small range of motion of the frying pan, enabled by the proposed training method. It is considered that the positional error in model A occurred to satisfy such constraints. In addition, since the switching between the static friction and the dynamic friction is a complicated problem to learn by end-to-end learning, the trained models could not reproduce the detailed movements.

Next, the performance with the pybullet simulation was evaluated. In this simulation, various disturbances exist that did not appear during training such as control deviations, vibrations in the z-axis, and rotational motions of both the object and the frying pan. Fig. 3.8 shows the reproduced trajectories by the three trained seq2seq models. In addition, Fig. 3.9 shows the snapshots of the manipulator for each seq2seq model. The length of the



(a) Object positions

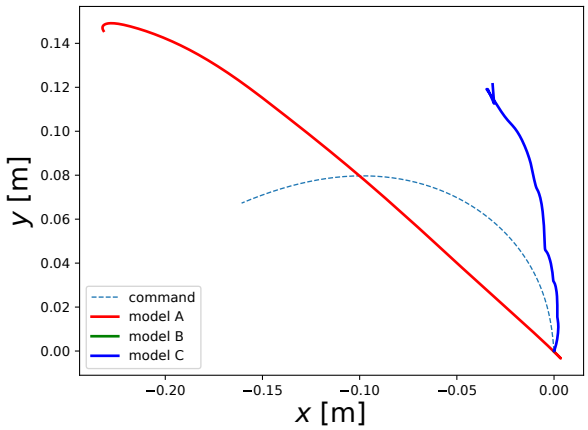


(b) Frying pan positions

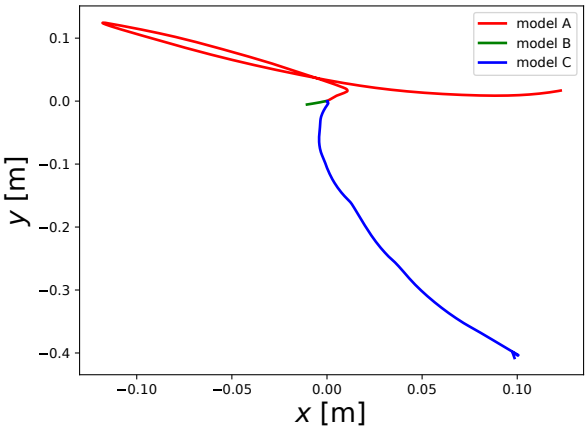
Figure 3.7: Reproduced trajectories of the object and the frying pan in an ideal situation.

command trajectory is set to $K = 1200$. Initially, from the result of model B, it is observed that both the object and the frying pan hardly moved. Thus, model B failed to learn the task. This is because the training losses vanished owing to the static friction as explained in Section 3.3. Meanwhile, model C succeeded in starting to slide the object owing to the training without the contact model. However, its trajectory did not follow the command trajectory. In addition, the manipulator extended the arm to the limit to reach a singular configuration and failed to perform the task. Because it trained without penalties such as (3.22), model C failed to satisfy the maximum range of motion. In contrast to these models, model A succeeded in sliding along the command trajectory. By switching to end-to-end learning including the penalty in (3.22), model A generated shaking motion of the frying pan within a small range. Thus, from these results, it is concluded that the proposed curriculum allowed the seq2seq models to learn nonprehensile manipulation tasks including the contact models.

Next, the generalization ability of the trained seq2seq models was verified by using the two-dimensional physics model with various friction parameters. Fig. 3.10 shows the



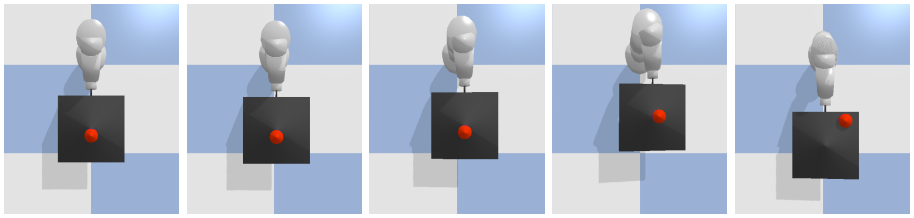
(a) Object positions



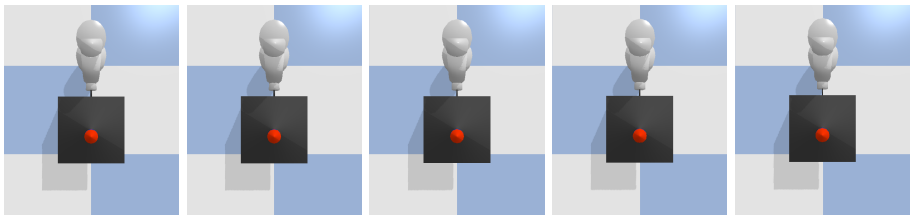
(b) Frying pan positions

Figure 3.8: Reproduced trajectories of the frying pan and the object in pybullet. In model C (blue curves), the manipulator extended the arm to the limit.

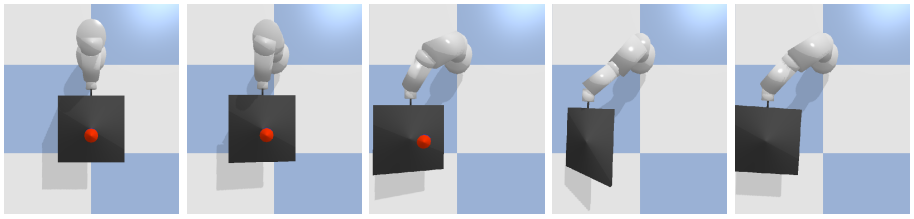
reproduction errors for various friction parameters. Here, the reproduction errors are calculated by the root-mean-squared errors between $\boldsymbol{p}_{\mathrm{d}}[0:K-1]$ and $\hat{\boldsymbol{p}}_{\mathrm{d}}[0:K-1]$. 10 trajectories for each parameter were calculated. It is noteworthy that the friction parameters were not provided to the seq2seq models. Model B resulted in large errors because the model could not move the object. Model C resulted in smaller errors than model B although the frying pan moved largely. In addition, the errors changed when the friction parameters changed. Meanwhile, model A resulted in errors as small as those of model C. Moreover, the errors hardly changed even if the friction parameters changed. From these results, it is concluded that the seq2seq model can handle parameter fluctuations that did not exist during training.



(a) Model A. The robot could manipulate the object.



(b) Model B. The robot did not move the frying pan.



(c) Model C. The robot dropped the object.

Figure 3.9: Snapshots for each trained seq2seq model.

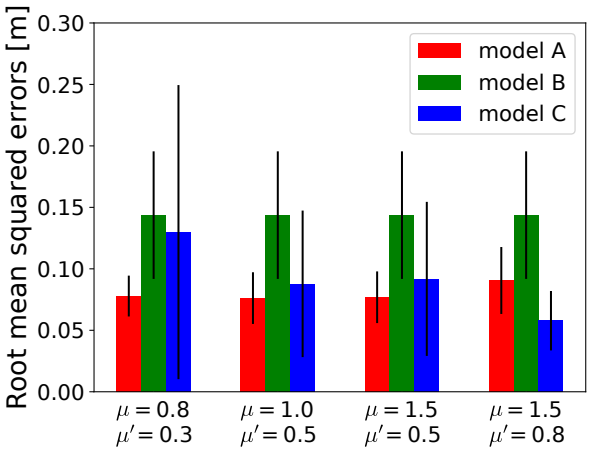


Figure 3.10: Reproduction errors of the object by the trained seq2seq models.

3.6 Conclusion

This chapter realized sequence-to-sequence (seq2seq) models for nonprehensile manipulation including nonlinear contact models to manipulate objects along the provided command trajectories. To handle the nonlinear contact models between the robots and objects, a training curriculum that commences training without the contact models are proposed. The proposed curriculum allows the seq2seq models to learn nonprehensile ma-

nipulation even if the contact models interfered with the backpropagation. The sliding manipulation was employed as an example. The proposed method was validated by using a manipulator in a dynamics simulator and two-dimensional physics models. In addition, it was observed that the trained seq2seq model could handle parameter fluctuations that were not provided during training. Such generalization ability suggests an advantage of the seq2seq models for complex tasks such as nonprehensile manipulation.

The future work is to investigate the reason that the proposed method makes neural networks robust against changes in physics parameters.

Chapter 4

Association of Latent Representations with External Orders by Mathematical Expressions[1]

In Chapters 2 and 3, training methods for seq2seq models to learn diverse trajectories were proposed. The decoders of the trained seq2seq models can be regarded as motion generators of diverse trajectories. Although the seq2seq models themselves can be used as motion generators that can receive trajectories as external orders, there is no guarantee that the deformation results satisfy the actual external orders. Hereinafter, the use of latent representations for diverse external orders are considered.

In this chapter, external orders that are expressed mathematically are taken. A backpropagation-based optimization method is proposed.

4.1 Introduction

As explained the previous chapters, it is generally difficult to generate motions that satisfy the dynamic constraint as well as accomplish the operational goals. Kinodynamic motion planning [53], which considers the dynamics and kinematics simultaneously, is a popular approach. Conventional studies have used various trajectory planning methods such as Rapidly-exploring Random Tree (RRT) [4, 37], Model Predictive Control (MPC) [1], and dynamic programming [3]. Such exploration methods, however, increase the calculation costs as the difficulty of the tasks increase. In addition, these methods tend to be difficult to apply to complex tasks such as contact motions.

To avoid the increase in complexity of trajectory optimization, approaches based on

[1]The contents in this chapter were published in: K. Kutsuzawa, S. Sakaino, and T. Tsuji, "Trajectory adjustment for nonprehensile manipulation using latent space of trained sequence-to-sequence model," *Adv. Robot.*, vol. 33, no. 21, pp. 1144–1154, oct 2019.

machine learning, especially neural networks, have been researched recently. Levine *et al.* [54] realized various assembly tasks including tight-fitting contact by using neural networks. Yuan *et al.* [55] realized a planar pushing task with obstacle avoidance by using reinforcement learning. Mordatch *et al.* [56] realized neural network-based feedback controllers that generate near-optimal walking motions. In addition, neural networks can also be used to reduce computational costs. Although neural networks require high computational costs during training, the trained models are computationally less expensive than most trajectory optimization methods. Zhang *et al.* [57] showed that neural networks trained by MPC can reduce the computation cost. Similarly, Furuta *et al.* [58] realized neural networks for dynamic manipulation by copying MPC, and confirmed that neural networks can generate appropriate trajectories faster than the original MPC.

Trajectory generation methods using machine learning, however, sometimes do not satisfy the task objectives. Since learning-based methods do not guarantee that the objective function will be satisfied, they sometimes generate trajectories beyond the dynamic constraint. Moreover, the objective changes when new obstacles appear or when the goal of the task changes. These issues also apply to the seq2seq models in this study. The seq2seq models themselves can be used as motion generators that can receive trajectories as external orders; however, there is no guarantee that the deformation results satisfy the actual external orders. In such cases, it is necessary to adjust the generated trajectories for the new objectives.

For adjusting the generated trajectories, it will be inefficient to optimize the trajectories directly, since the kinodynamic motion planning problem is difficult to solve. Although there have been several researches on trajectory adjustment [38,39,41,42], all of them used domain-specific algorithms. In this chapter, on the other hand, intermediate representations of trained neural networks, which are also called latent representations are used. Since the latent representations express task-specific features in a low-dimensional space, they are expected to be optimized by simple methods. The effectiveness of the use of latent representations is demonstrated in the field of computer vision. The use of gradient descent methods to generate images resulted in unrealistic images [59], whereas the use of gradient descent methods to latent representations of image classification models could obtain realistic images [60]. Recently, some studies used latent representations of motions in the field of reinforcement learning [11, 13, 61]. Our proposed method, on the other hand, can optimize the latent representations directly by using gradient descent methods for various objective functions. Moreover, the proposed method is capable of extending the performance and application of the trajectory deformation models, which enables it to perform dynamic manipulation even by itself. Note that some studies in robotics focused on the optimization of internal representations by backpropagation [62, 63].

This chapter proposes a method of adjusting the trajectories generated by seq2seq models by using the latent representations of the trajectories. By optimizing their latent representations, the solution can be obtained in fewer iterations than that required when optimizing the trajectories directly, as shown in the simulation results. This method

can be used for various objective functions; it is applicable when the objective functions are differentiable. In addition, the latent representations can be optimized by using the simplest gradient descent method.

The rest of this chapter is organized into the following sections. Section 4.2 explains the proposed method. Section 4.3 describes the simulation that is conducted to evaluate the proposed method. Section 4.4 describes an experiment based on the proposed method. Finally, Section 4.5 concludes this chapter.

4.2 Method

4.2.1 Issue to be Addressed

In the trajectory deformation using seq2seq models, the training objective is to minimize the amount of deformation and retain the dynamic constraint. However, there often exist different objectives when using trained models. For example, users often desire to reach certain end positions. In such cases, the objective is to minimize the error at the end of the trajectories, while the remaining trajectories can be ignored. For example, the velocity of the deformed trajectories may be larger than what the robots can follow accurately. In such cases, the velocity should be decreased slightly while satisfying the dynamic constraint.

Therefore, users often have to adjust the deformed trajectories for the given objective according to the situation. In this section, a method of adjusting the deformed trajectories by utilizing the architecture of seq2seq models is proposed.

4.2.2 Optimization of the Latent Representation

The most straight-forward approach is to optimize the trajectories directly with the given objective functions. There have been many studies on trajectory adjustment [38,39, 41,42]; however, this approach may be difficult because of the large number of dimensions.

Instead of utilizing this approach, latent representations can be used for optimization. The latent representations represent the features of the trained time-series in a low-dimensional space. In this task, the latent representations are expected to represent the trajectories that satisfy the dynamic constraint. Therefore, it is expected that the use of latent representations instead of the trajectories are beneficial to such adjustment tasks.

To optimize the latent representations, it is required to quantify the relationship between the latent representations and the deformed trajectories. This relationship is non-obvious; however, their small displacements can be associated using Jacobian matrices. Since backpropagation can be applied to the trained decoder, the derivative function $\frac{\partial U_{\text{out}}}{\partial \boldsymbol{z}}$ can be obtained. Therefore, the gradient of the objective function J with respect to the latent representations $\boldsymbol{z}$ can be obtained as follows:

$$\frac{\partial}{\partial \boldsymbol{z}} J(X_{\text{out}}, U_{\text{out}}) = \frac{\partial U_{\text{out}}}{\partial \boldsymbol{z}} \frac{\partial}{\partial U_{\text{out}}} J(X_{\text{out}}, U_{\text{out}}). \tag{4.1}$$

Finally, the objective function can be optimized with gradient-based optimization methods such as gradient descent as follows:

$$\boldsymbol{z} \leftarrow \boldsymbol{z} - \eta \frac{\partial}{\partial \boldsymbol{z}} J(X_{\text{out}}, U_{\text{out}}), \tag{4.2}$$

where η is the learning coefficient.

The objective functions $J(X_{\text{out}}, U_{\text{out}})$ usually differ from the loss function used in the training of seq2seq models. Although the changes in the objective functions between training and testing generally cause poor performance, the proposed method is expected to work due to the difference in optimization targets; the training of seq2seq models optimizes their connection weights, while the proposed method optimizes the latent representations. Besides, the proposed method requires that the latent space represents various motions satisfying the dynamic constraint, which can be accomplished by training with various input trajectories. Thus, the proposed method works in various objective functions as long as the dynamic constraint is the same and the seq2seq models are trained with various trajectories.

The proposed method has another advantage; thanks to the optimization of the latent space, it is considered to be robust against the hyperparameters of the seq2seq models. Even if the seq2seq models cannot achieve the minimum training loss due to the hyperparameters, such small performance changes can be relieved by iterating the optimization steps in (4.2).

4.2.3 Optimization Procedure

The proposed method progresses as follows:

step 1 Prepare an original trajectory X_{in}.

step 2 Input X_{in} to the encoder and obtain $\boldsymbol{z}$.

step 3 Generate trajectories X_{out} and U_{out} from $\boldsymbol{z}$.

step 4 Calculate the objective function $J(X_{\text{out}}, U_{\text{out}})$.

step 5 Calculate the gradient $\frac{\partial}{\partial \boldsymbol{z}} J(X_{\text{out}}, U_{\text{out}})$ by backpropagation as in (4.1).

step 6 Update $\boldsymbol{z}$ by using (4.2).

step 7 Repeat 3–6 until convergence.

Finally, an optimized $\boldsymbol{z}$ and its corresponding trajectory X_{out} can be obtained. An overview of this procedure is illustrated in Fig. 4.1.

4.3 Simulation

In this chapter, the same task as Chapter 2, turning over pancakes with a spatula in planar physics, is employed.

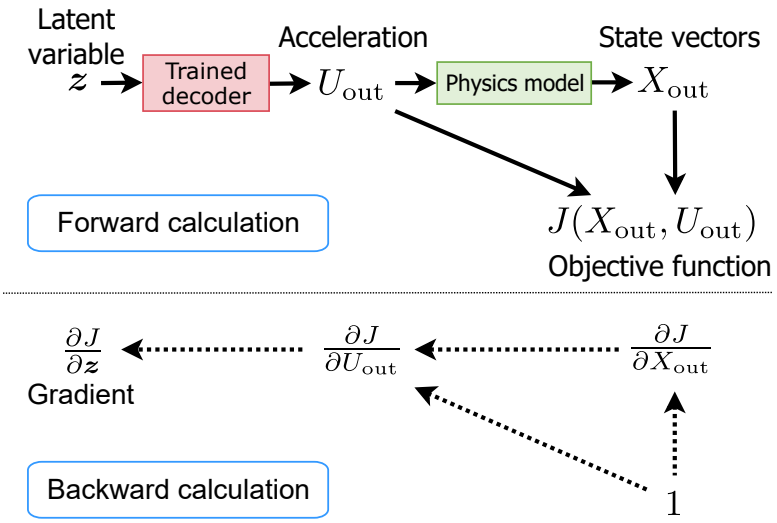


Figure 4.1: Overview of the proposed method. Top: the forward calculation graph. Bottom: the backward calculation graph.

4.3.1 Implementation of Seq2seq Model

The training procedure and the model architecture followed those in Chapter 2; however two layers of LSTMs with 64 units were used as the encoder and the decoder. Thus, the latent representation $\boldsymbol{z}$ has $64 \times 2 = 128$ dimensions. It is more than 10 times smaller than that of the trajectories; as explained in Section 4.3.2, each trajectory has $3 \times 500 = 1500$ dimensions.

1638400 input trajectories are used to train the seq2seq model. These trajectories are generated in the simulation.

4.3.2 Adjusting Trajectories to Reach the Given End Positions

Trajectories for reaching the given end positions were generated by using the trained seq2seq model.

Objective function

To evaluate the objective, the following objective function was designed:

$$J_1(X_{\text{out}}, U_{\text{out}}) \equiv \|\boldsymbol{D}(\boldsymbol{p}[K-1] - \boldsymbol{p}_{\text{end}})\|^2 + \alpha f_{\text{dyn}}(X_{\text{out}}, U_{\text{out}}), \tag{4.3}$$

where, $\boldsymbol{p}_{\text{end}}$ denotes the desired end position, and α is a constant value; here, it was set to 1. $\boldsymbol{D} = \text{diag}[1, 1, 0.1]$ is a scaling parameter between the position [m] and the attitude [rad]. f_{dyn} denotes the penalty term for the dynamic constraint, which is defined as follows:

$$f_{\text{dyn}}(X_{\text{out}}, U_{\text{out}}) \equiv \sum_{k=0}^{K-1} F_{\text{c}}[k]. \tag{4.4}$$

Here, $F_{\mathrm{c}}[k]$ is detailed in (2.18). This penalty increases when (2.6) is not satisfied. The length of a sequence K is set to $K = 500$.

The above objective function includes f_{dyn}, the same penalty term as used during training. Even if the seq2seq models are trained correctly, such a penalty is still required to satisfy the dynamic constraint. The reason is that not all points in the latent space are associated with valid trajectories. Latent representations work correctly only if the encoder generated them during training; otherwise, there is no guarantee that these are associated with valid trajectories. Even though such valid latent representations have a dense and continuous distribution in a certain region, invalid latent representations that were not generated by the encoder during optimization can be found.

Various patterns of end positions $\boldsymbol{p}_{\mathrm{end}} = [y_{\mathrm{end}}, z_{\mathrm{end}}, \theta_{\mathrm{end}}]$ were sampled from the following range:

$$-0.1 \text{ m} \leq y_{\mathrm{end}} \leq 0.1 \text{ m}, \tag{4.5}$$

$$-0.1 \text{ m} \leq z_{\mathrm{end}} \leq 0.1 \text{ m}, \tag{4.6}$$

$$-\pi \text{ rad} \leq \theta_{\mathrm{end}} \leq \pi \text{ rad}. \tag{4.7}$$

Three points at equal intervals from each axis were sampled. In total, 27 points of end positions were used.

Original trajectories

As the inputs to the seq2seq model, trajectories satisfying the following objective function were prepared:

$$\hat{J}_1(X_{\mathrm{out}}, U_{\mathrm{out}}) \equiv \|\boldsymbol{D}(\boldsymbol{p}[K-1] - \boldsymbol{p}_{\mathrm{end}})\|^2 \tag{4.8}$$

This is the first term of (4.3). Since this objective function considers only the kinematics, it is easy to design. S-curve acceleration/deceleration trajectories from the initial position to the end position were used.

Results

The objective function J_1 was minimized by using the trained seq2seq model. The following three methods were compared:

case 1 Updating the latent representation $\boldsymbol{z}$ starting with the encoder result (*i.e.*, the proposed method).

case 2 Updating the latent representation $\boldsymbol{z}$ starting with a random value.

case 3 Updating the output trajectory of the decoder, U_{out}, starting with the decoder result.

The comparison between **case 1** and **case 3** shows that optimization of the latent representations is better than the optimization of the trajectories. **Case 2** can verify whether

the use of decoder alone can achieve the objective, since it does not use the encoder. In **case 2**, the initial value of the latent representation $\boldsymbol{z}_0$ was sampled from the Gaussian distribution with mean $\boldsymbol{0}$ and variance $\boldsymbol{I}$.

In general, the distribution of the latent representations is not a standard Gaussian distribution. The actual distribution may be complex and is hard to identify. Therefore, in **case 2**, the standard Gaussian distribution was used. Even though seq2seq models can be extended to specify the distribution of the latent representations [64], such models often fail to learn various trajectories.

The progress of the objective function values is shown in Fig. 4.2. Here, the learning coefficients were set to $\eta = 200, 200,$ and 1000 for **case 1**, **case 2**, and **case 3**, respectively. In **case 1**, the objective function value decreased quickly. The values decreased below 10^{-3} after the 5th update on average. In **case 2** also, the values decreased. However, the initial loss was larger than that in the other cases. In addition, the convergence speed was slower than that of **case 1**. The reason can be considered that the initial latent representations were far from the optimal one. In **case 3**, the objective function value hardly decreased. Here, 10 updates took 12.1 s (*i.e.*, 1.21 s/update) in **case 1** and **case 2**, and 10.5 s (*i.e.*, 1.05 s/update) in **case 3**, with Intel Core i7-8700. Although the back-propagation path includes the decoder in **case 1** and **case 2**, the calculation time was only 15 % longer. Although the calculation times were longer than those of the trained neural networks, it is notable that this method did not require additional training for the novel objective. Therefore, considering the cost of training new models due to changes in the objective, the proposed method is computationally efficient.

The trajectories obtained after 100 updates are shown in Figs. 4.3, 4.4, and 4.5. All the trajectories reached the given end position with little errors. In addition, the dynamic constraint was maintained in all cases. The root mean square errors (RMSEs) of the end positions are shown in Fig. 4.6. The RMSEs in **case 1** were 0.27 mm and 0.30 deg on average. These results were comparable to the neural networks in [58], which were trained with thousands of training samples and 100000 iterations. On the other hand, **case 2** and **case 3** resulted in larger errors, especially in the attitude. Since the turning-over task, which is performed by tilting the spatula largely, is difficult, it is considered that the optimized trajectories result in large errors in the attitude unless using a low-dimensional latent space and starting from good initial values.

4.4 Experiments

4.4.1 Setup of the Robot

A six DOF "MOTOMAN-MH3F," supplied by Yaskawa Electric, was used. A spatula was equipped at the tip of the manipulator. Instead of a pancake, a rubber plate was placed on the spatula.

A P-D controller with DOB was implemented to control the tip of the spatula position and attitude. The control system is illustrated in Fig. 4.7. Here, $\boldsymbol{q}$ indicates the six-

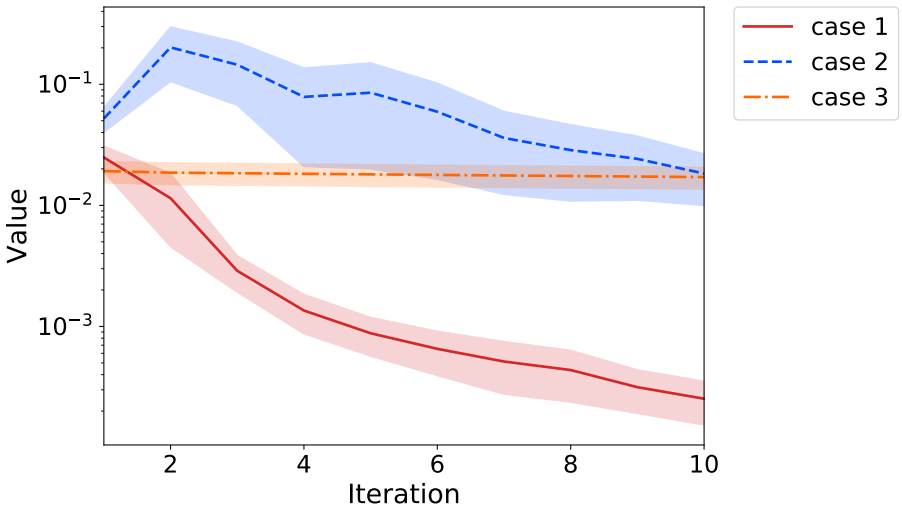


Figure 4.2: Progress of the objective functions. The lines indicate the mean losses of 27 trials with various end positions. The filled areas indicate their 95 % confident levels.

dimensional position/attitude of the spatula in the Cartesian coordinate system. The control command, $\boldsymbol{q}^{\mathrm{cmd}}$, was calculated by the position generated by the seq2seq models, $\boldsymbol{p}[k] = [y[k], z[k], \theta[k]]^\top$, as follows:

$$\boldsymbol{q}^{\mathrm{cmd}} = \left[0.5, y[k], z[k] + 0.35, \theta[k], \frac{\pi}{2}, 0\right]^\top. \tag{4.9}$$

Here, the control system did not receive the generated acceleration $\boldsymbol{u}[k]$ directly to avoid drift errors, which cause large errors in the final positions. Even though the control commands were the position and attitude, the DOB forced the robot to follow the acceleration reference, $\ddot{\boldsymbol{q}}^{\mathrm{ref}}$, in a frequency range lower than the cutoff frequency g_{DOB}. $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$ indicate the joint angles and joint torques, respectively, $\bullet^{\mathrm{res}}$ and $\bullet^{\mathrm{ref}}$ indicate response values and reference values, respectively, and $\hat{\boldsymbol{\tau}}^{\mathrm{dis}}$ indicates the disturbance torque estimated by the DOB. $\boldsymbol{K}_{\mathrm{p}} = \mathrm{diag}[K_{\mathrm{pp}}, K_{\mathrm{pp}}, K_{\mathrm{pp}}, K_{\mathrm{pr}}, K_{\mathrm{pr}}, K_{\mathrm{pr}}]$, $\boldsymbol{K}_{\mathrm{d}} = \mathrm{diag}[K_{\mathrm{dp}}, K_{\mathrm{dp}}, K_{\mathrm{dp}}, K_{\mathrm{dr}}, K_{\mathrm{dr}}, K_{\mathrm{dr}}]$, $\boldsymbol{J}$, and $\boldsymbol{M}$ indicate the proportional gain, derivative gain, Jacobian matrix, and mass matrix, respectively. The proportional position gain K_{pp}, proportional attitude gain K_{pr}, derivative position gain K_{dp}, and derivative attitude gain K_{dr} were 450, 700, 60, and 90, respectively. The cutoff frequency of the disturbance observer g_{DOB} was set to 2.0 Hz. The control period was 1 ms.

4.4.2 Trajectory

While executing the turning over motions, the manipulator may fail in the task if the velocity is too large. Therefore, the trajectories should be adjusted to reduce the velocity.

To obtain such low-velocity trajectories, the following objective function is used:

$$J_2(X_{\text{out}}, U_{\text{out}}) \equiv \max\left(\sqrt{y^2[K-1] + z^2[K-1]} - p_{\text{lim}}, 0\right) + 0.1 \max\left(|\theta[K-1] - \theta_{\text{end}}| - \theta_{\text{lim}}, 0\right) + \alpha f_{\text{dyn}}(X_{\text{out}}, U_{\text{out}}) + \frac{\beta}{K} \sum_{k=0}^{K-1} \| \max(0, \boldsymbol{D}(\text{abs}(\dot{\boldsymbol{p}}[k]) - \boldsymbol{v}_{\text{lim}}))\|^2. \quad (4.10)$$

Here, abs indicates the element-wise absolute value function. θ_{end}, p_{lim}, and θ_{lim} indicate the desired end attitude, desired range of the end position, and attitude, respectively; these parameters were set to $-\frac{3}{4}\pi$ rad, 0.2 m, and 0.5 rad, respectively. α and β are constant values; here they were set to $\alpha = 1$ and $\beta = 0.01$. $\boldsymbol{v}_{\text{lim}}$ indicates the desired range of the velocity; it was set to $[1.5, 1.5, 4.5]^\top$. In this objective function, the first two terms indicate the positional constraint. These terms increases when the errors exceed the given range. The final term indicates the velocity penalty. This term increase when the velocity exceeds $\boldsymbol{v}_{\text{lim}}$. For optimization, the learning coefficient was set to $\eta = 100$.

The input trajectory to the seq2seq model had a uniform linear motion with constant velocity from the initial position to the end position $\boldsymbol{p}_{\text{end}} = \left[0, 0, -\frac{3}{4}\pi\right]^\top$.

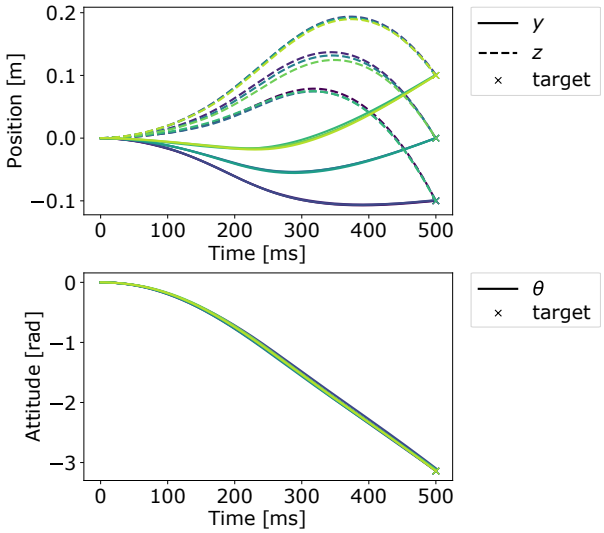
4.4.3 Result

The optimization results in the 0th, 10th and 20th iterations are shown in Figs. 4.8 and 4.9. Before optimization (*i.e.*, the 0th iteration), the velocity was larger than 2 m/s and 2π rad/s, and the dynamic constraint was not maintained. As the update of the latent representation progressed, the velocity and ϕ decreased. Finally, a trajectory with a lower velocity was obtained.

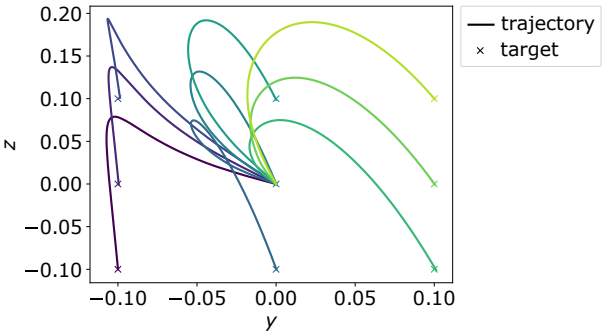
The final trajectory at the 20th iteration was executed by the manipulator. The experimental result is shown in Fig. 4.10. The motion started from 25 s and lasted 0.5 s. After the turning over motion, a constant deceleration motion with 0.1 s was added to avoid large deceleration in the command trajectory. Although there remained some control deviations, the manipulator succeeded in the turning over motion as shown in Fig. 4.11.

4.5 Conclusion

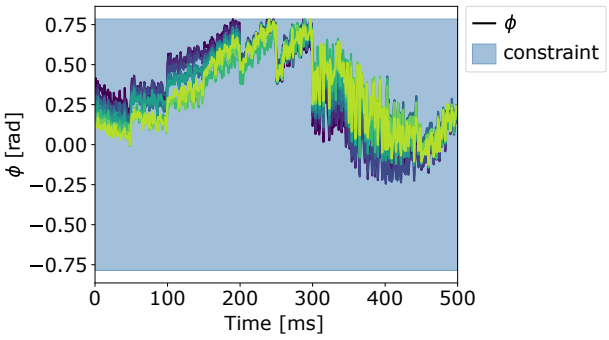
In this chapter, a method of adjusting the the latent representations of seq2seq models was proposed. The proposed method optimizes the latent representations instead of trajectories to minimize the given objective functions. Through simulation, it was verified that the use of latent representations can obtain the desired trajectories faster than that when optimizing the trajectories directly. In addition, it was confirmed that the trajectories adjusted by the proposed method can be executed by an actual manipulator.



(a) Time-position

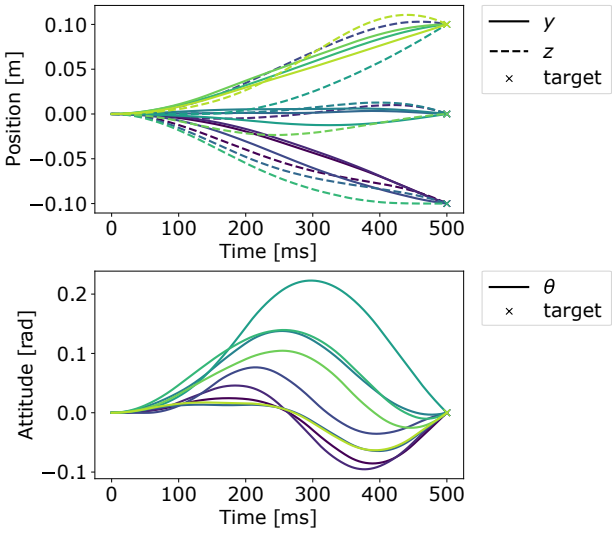


(b) Trajectories in the y-z plane. The x-marks indicate the target positions.

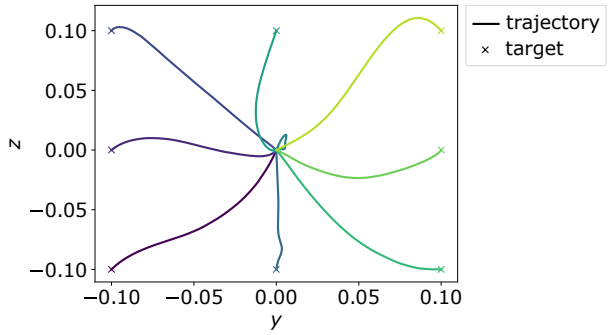


(c) Angle of the contact force of the pancake with respect to the spatula. The dynamic constraint is satisfied in the blue area.

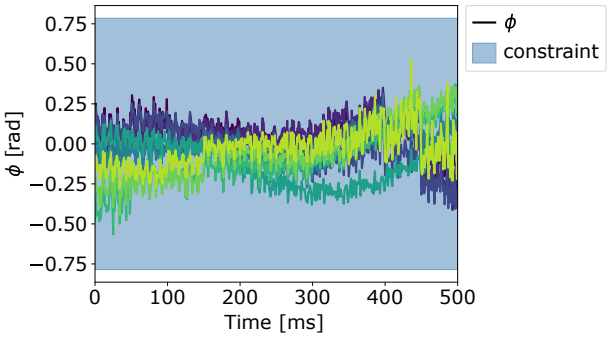
Figure 4.3: Trajectories obtained after optimization by the proposed method when the final attitude is $-\pi$.



(a) Time-position

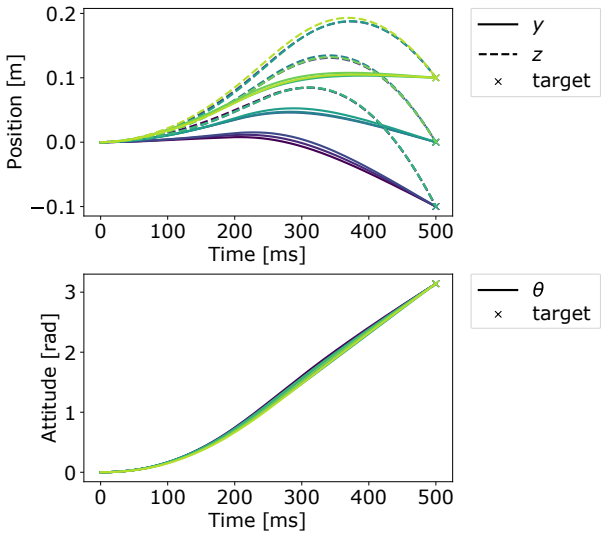


(b) Trajectories in the y-z plane. The x-marks indicate the target positions.

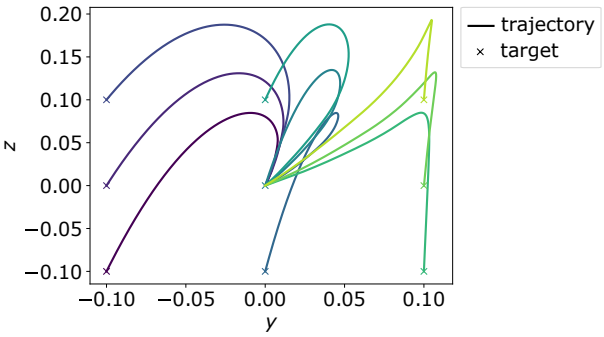


(c) Angle of the contact force of the pancake with respect to the spatula. The dynamic constraint is satisfied in the blue area.

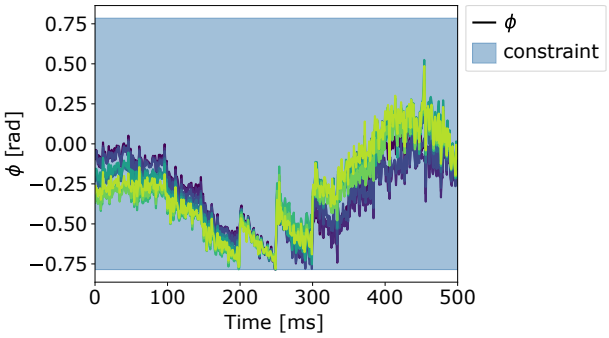
Figure 4.4: Trajectories obtained after optimization by the proposed method when the final attitude is 0.



(a) Time-position



(b) Trajectories in the y-z plane. The x-marks indicate the target positions.



(c) Angle of the contact force of the pancake with respect to the spatula. The dynamic constraint is satisfied in the blue area.

Figure 4.5: Trajectories obtained after optimization by the proposed method when the final attitude is π.

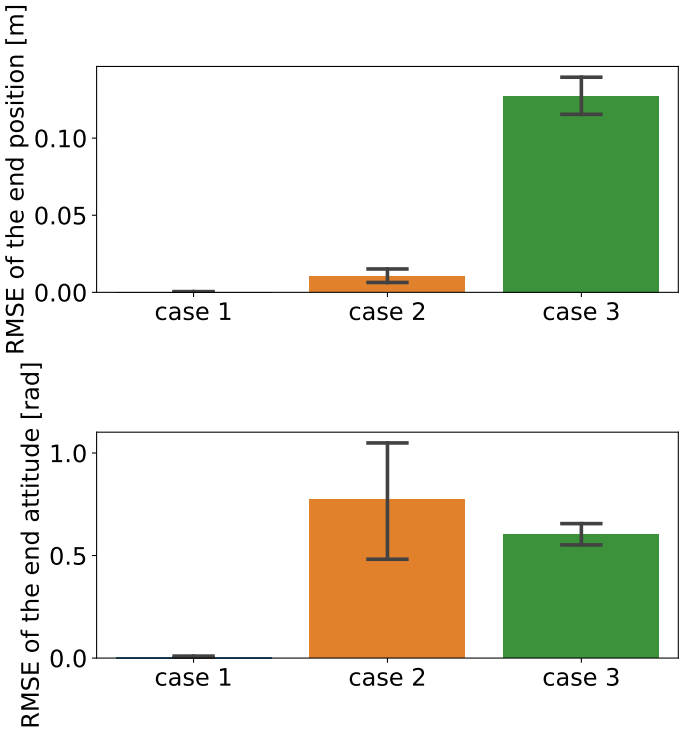


Figure 4.6: Root mean square errors (RMSEs) of the end positions. The error bars indicate their 95 % confident levels.

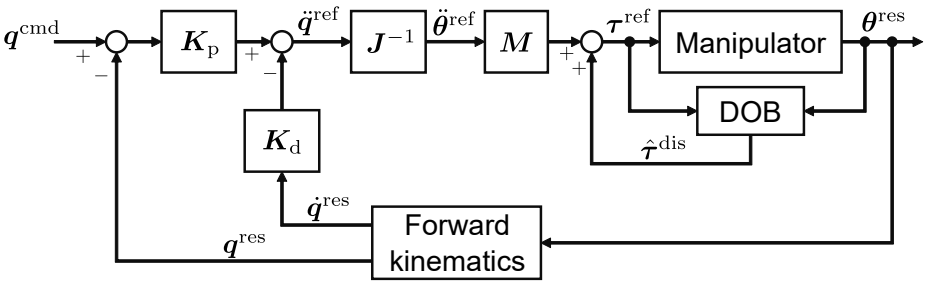
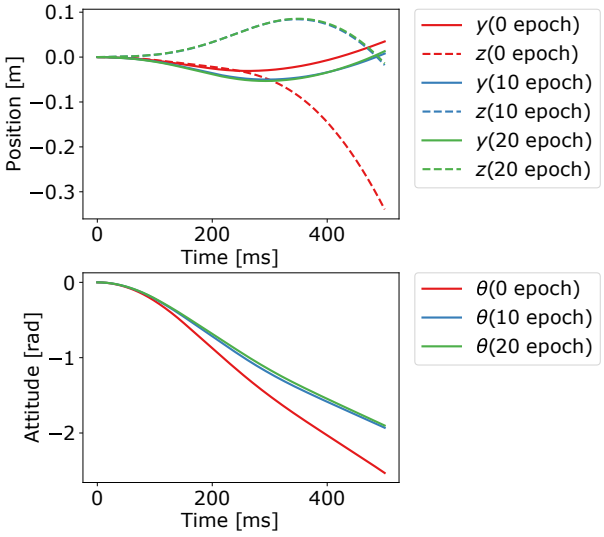
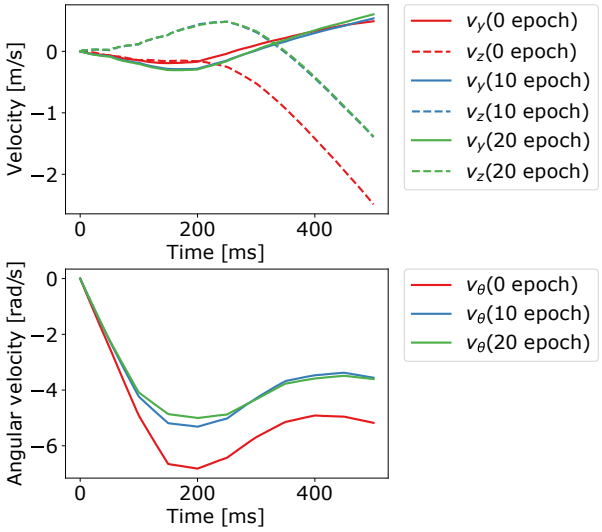


Figure 4.7: Control system.

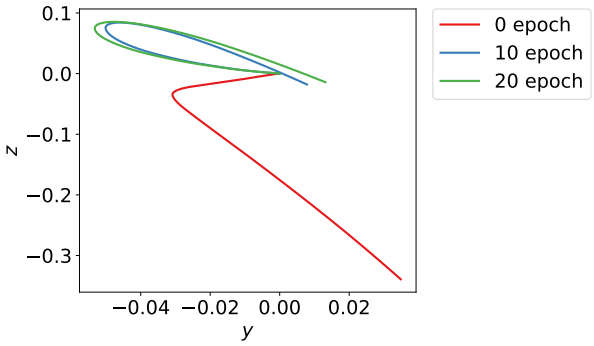


(a) Time-position

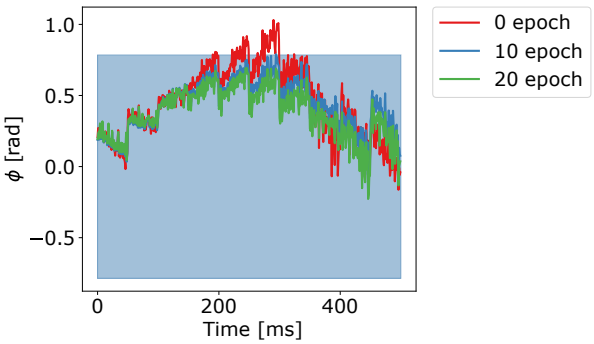


(b) Time-velocity

Figure 4.8: Trajectories obtained after optimization with J_2.

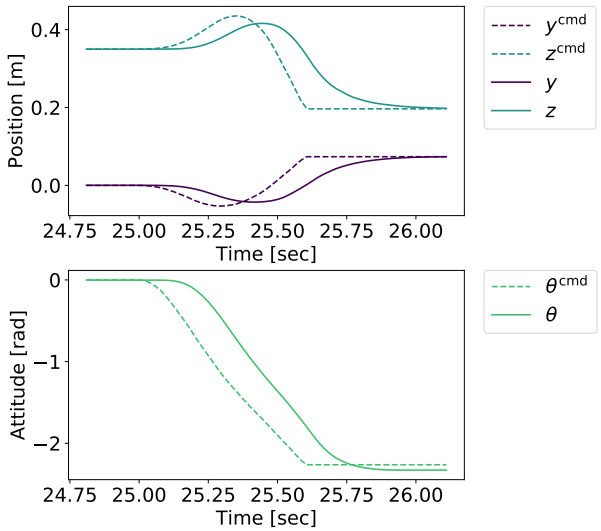


(a) Trajectories in the y-z plane.

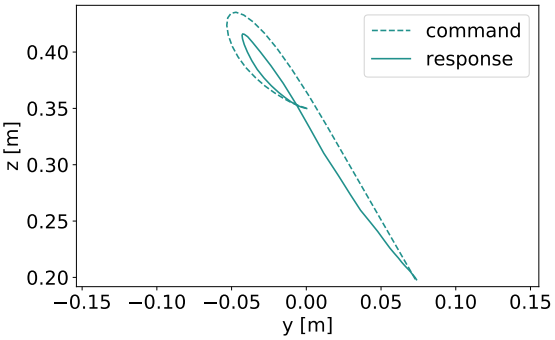


(b) Angle of the contact force of the pancake with respect to the spatula. The dynamic constraint is satisfied in the blue area.

Figure 4.9: Trajectories obtained after optimization with J_2.



(a) Position and attitude.



(b) Trajectories in the y-z plane.

Figure 4.10: Trajectories obtained after optimization by the proposed method with $\boldsymbol{p}_{\mathrm{end1}}$.

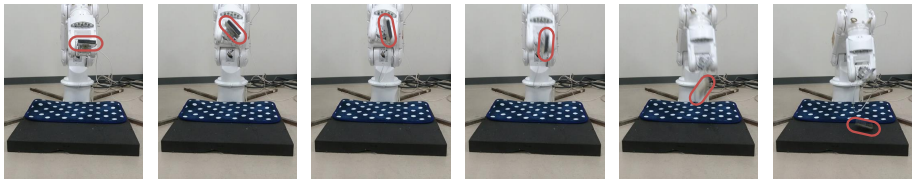


Figure 4.11: Snapshots of the turning over motion.

Chapter 5

Association of Latent Representations with External Orders by Numerical Expressions

The final topic in this study is learning from demonstration. By enabling to train motion generators using human demonstrations, the applications are expected to be extended.

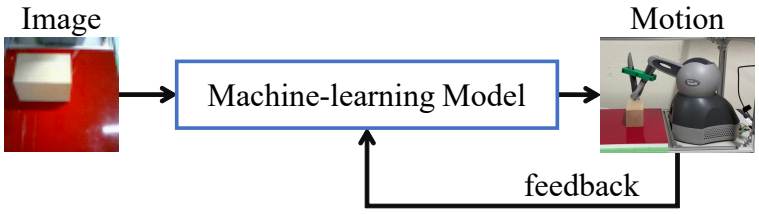
In this chapter, external orders that are expressed by demonstration data are taken. A training method of associating the demonstration data and latent representations is proposed.

5.1 Introduction

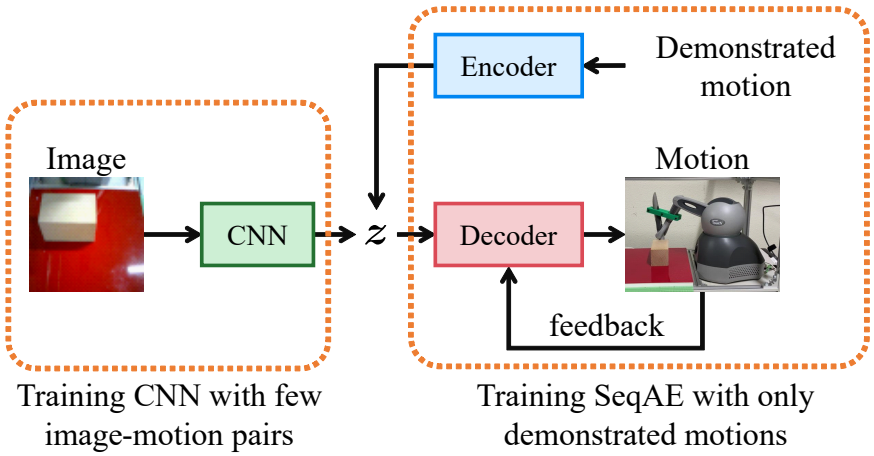
Skill transfer from human experts to robots is beneficial for such tasks. Because human experts are usually not specialists in robotics, *Learning from Demonstration* (LfD) [65,66] that trains robots by demonstrating skills performed by human experts, is an effective approach. In addition, the amount of training data required by LfD is smaller than those required by reinforcement learning and self-supervised learning, both of which include exploration phases during training. Neural networks are being more widely used to acquire complex skilled motions [33].

Many robotics tasks require visual information. For example, cooking tasks require vision to detect the positions of foods or tools. Robots must generate motions (*i.e.*, joint angles, velocity, torques, *etc.*) based on the images that they capture. Therefore, in LfD, the robots learn the image–motion relationship as illustrated in Fig. 5.1(a). Note that this study considers the case that a single image corresponds to a whole motion.

LfD with visual inputs, however, has some issues. First, visual conditions can easily change due to many factors that are not directly related to the task, such as camera positions, light conditions, and background colors. When the visual conditions significantly changed, the same images can no longer be used for training. Considering that training



(a) Online motion generation by a machine-learning model that learned image–motion relationship



(b) Approach of this study; using SeqAE for improving the robustness of online motion generation. Here, $\boldsymbol{z}$ indicates the latent representation.

Figure 5.1: Overview of this study

data are collected by human experts, recollecting a large amount of demonstrations must be avoided.

Another issue is caused by the training method used for motion generation. In many cases, machine-learning models generate control commands based on the current states of the robot and the environment. These control commands affect the next states of the robot and the environment, and finally, the response values are fed back to the models. Therefore, the motion generation process is auto-regressive in that it configures a feedback loop that includes the dynamics of the robot and the environment. During the training of models, however, such dynamics are neglected. The models typically receive training samples instead of the actual response values because the actual robots cannot be used during training. This method is called *teacher forcing* [26], which does not need to actuate actual robots during training. Fluctuations in model outputs and environmental uncertainty do not accumulate in teacher forcing; in contrast, during the operation phase, such fluctuations are fed back to the models and they accumulate. If the models are not robust against such fluctuations, the task performance degrades [67–69]. This issue is referred to as *exposure bias* [24, 25]. To achieve model robustness against such fluctuations, a large amount of demonstrations is necessary.

To address these issues, it is effective to use demonstrated motions that are not associated with images to improve the robustness of the motion generation. Such motions are obtained when the visual conditions change; generally, they cannot be used for the image-motion-relationship training. Instead of associating such motions with images, these motions can be associated with low-dimensional latent representations.

Recently, many studies have focused on latent representations. They can be used for semi-supervised learning that improves the model performance by using a large amount of unlabeled data [70, 71]. In LfD, latent representations of images are often used for dimensionality reduction of images [72, 73]. Latent representations of motions are also utilized mainly for reinforcement learning for reducing the exploration costs by dimensionality reduction [74, 75]. Latent representations of motions can also be used for editing motions [7, 9, 61].

In this study, a method that uses sequence-to-sequence autoencoders (SeqAEs) [15] to improve the performance of LfD with visual inputs is proposed. An overview of this approach is illustrated in Fig. 5.1(b). SeqAEs are seq2seq models that reconstruct input time-series. Some studies on natural language processing have demonstrated that the use of SeqAEs improves sequence-generation performance [76, 77]. Therefore, in this study, these method is applied to robots for motion generation. In addition, even when the visual conditions change, users only need to retrain part of the model because the latent representations of motions can be reused.

The remainder of this chapter is organized as follows. In Section 5.2, a method of using SeqAE for LfD with visual inputs is proposed. The proposed method is evaluated in Section 5.3. Finally, the conclusions of this study is presented in Section 5.4.

5.2 Method

5.2.1 LfD using SeqAE

As described in Section 5.1, machine-learning models are susceptible to exposure bias because the motion generation includes the dynamics of the robots and the environment. Although a large amount of image-motion pairs are necessary to improve the robustness of the models, it is difficult because the visual conditions easily change.

To address these issues, the training process is to be divided into three phases: 1) training motions to a SeqAE to obtain a robust motion-generation model that does not depend on images, 2) training a convolutional neural network (CNN) to associate images with latent representations obtained by the SeqAE, and 3) combining them to construct a neural model that generates motions based on images. An overview of the method is illustrated in Fig. 5.2. The procedure is as follows.

Training SeqAE

First, a SeqAE is trained with demonstrated motions. Here, it is considered that the demonstrated motion consists of time series of joint angles $\boldsymbol{\theta}(t)$, joint angular velocity $\dot{\boldsymbol{\theta}}(t)$, and external torque $\boldsymbol{\tau}(t)$ of a robot. Note that this method can be applied even when the motions consist of other variables; for example, in Section 5.3, motions include all variables in a master-slave system.

Input/output of the SeqAE will be designed according to the motion data and control system. The encoder should receive all variables in the motions to obtain latent representations of motions. Besides, for online motion generation, the decoder should receive the current state variables and generate the next command values to the control system. In the above case, the encoder receives time series of the motion, $\boldsymbol{\theta}(t)$, $\dot{\boldsymbol{\theta}}(t)$, and $\boldsymbol{\tau}(t)$, and finally generates a latent representation $\boldsymbol{z}$; the decoder receives $\boldsymbol{z}$ and the current state, $\boldsymbol{\theta}(t)$, $\dot{\boldsymbol{\theta}}(t)$, and $\boldsymbol{\tau}(t)$, and generates the next state, $\hat{\boldsymbol{\theta}}(t+\Delta t_{\mathrm{NN}})$, $\hat{\dot{\boldsymbol{\theta}}}(t+\Delta t_{\mathrm{NN}})$, and $\hat{\boldsymbol{\tau}}(t+\Delta t_{\mathrm{NN}})$. These outputs will be used as the command values for a position/force control system of the robot. Here, Δt_{NN} indicates the activation interval of the decoder; $\hat{\bullet}$ indicates the predicted values. A loss function of the SeqAE is expressed as follows:

$$\begin{aligned} L_{\mathrm{motion}} =& \frac{1}{\sigma_{\theta}^2}\sum_{k=1}^{K}\left\|\hat{\boldsymbol{\theta}}(k\Delta t_{\mathrm{NN}})-\boldsymbol{\theta}(k\Delta t_{\mathrm{NN}})\right\|^2 \\ &+\frac{1}{\sigma_{\dot{\theta}}^2}\sum_{k=1}^{K}\left\|\hat{\dot{\boldsymbol{\theta}}}(k\Delta t_{\mathrm{NN}})-\dot{\boldsymbol{\theta}}(k\Delta t_{\mathrm{NN}})\right\|^2 \\ &+\frac{1}{\sigma_{\tau}^2}\sum_{k=1}^{K}\left\|\hat{\boldsymbol{\tau}}(k\Delta t_{\mathrm{NN}})-\boldsymbol{\tau}(k\Delta t_{\mathrm{NN}})\right\|^2. \end{aligned} \tag{5.1}$$

Here, $\sigma_{\bullet}^2$ indicates the variance over the dataset; it is used as a scaling factor. K indicates the maximum index in the time series of the demonstration.

In the training of the SeqAE, the demonstrated motions do not need to be associated with images. Therefore, once collecting a set of demonstrations, they can be reused even when visual conditions change. By training the SeqAE with a larger amount of motions, the decoder will be more robust against fluctuations of the states from the demonstrations.

Training CNN

A CNN is trained to associate images with the latent representations obtained by the encoder of the trained SeqAE. The CNN receives images and predicts latent representations of motions $\boldsymbol{z}$ that were associated with these images. Due to the feature extraction by SeqAEs, the number of output dimensions of the CNN are reduced. In addition, because the output of the CNN is not a time series but single variables, the training is expected to be stable even if the number of images is small.

A loss function of the CNN is expressed as follows:

$$L_{\mathrm{image}} = \|\boldsymbol{z}-\hat{\boldsymbol{z}}\|^2. \tag{5.2}$$

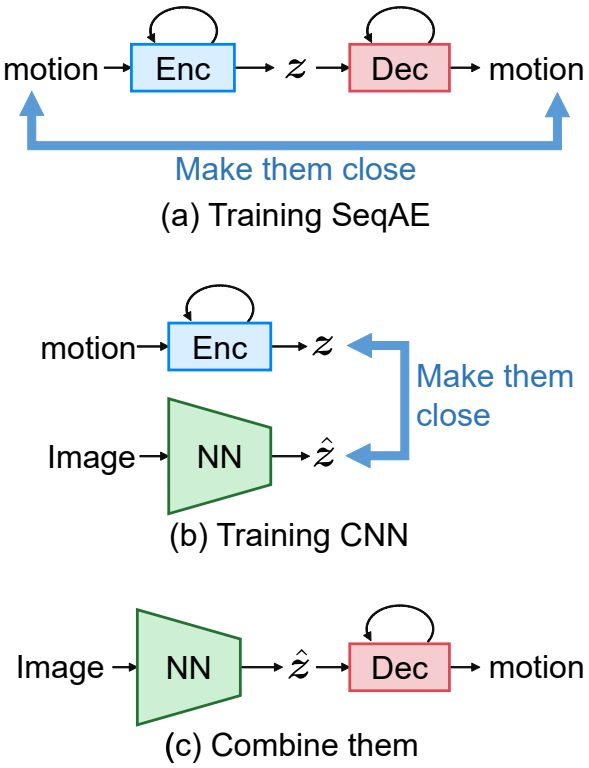


Figure 5.2: Overview of the proposed method

Here, $\boldsymbol{z}$ indicates a latent representation of a motion generated by the encoder of the trained SeqAE; $\hat{\boldsymbol{z}}$ indicates an output of the CNN.

Combine trained models

By making $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$ close, motions can be generated by using both the demonstrated motions and the images. By replacing the encoder in SeqAE to the trained CNN, a neural model that generates motions according to given images is obtained.

5.3 Experiments

5.3.1 Task Specification

As an example, the task of pushing a wood block that was 10 cm wide, 5 cm high, and 5 cm deep was chosen. The overview of the task setup is shown in Fig. 5.3. In this task, a webcam takes a 64×64 pixel picture just before the task, then a robot pushes the block based on its position. When the block is on the left side, the robot pushes it to the right; and when it is on the right side, the robot pushes it to the left side. The task was regarded as successful when the center of the block was moved across the center of the robot. An example of the demonstrations is shown in Fig. 5.4. Examples of the images are shown in Fig. 5.5.

Although this task is a toy-problem, it includes two important features common in many practical tasks such as assembly and cooking: 1) the robot needs to detect the object positions from the images and 2) the task includes contact with objects and friction between the objects and the environment that are difficult to model. Thus, the robot should learn this task through human demonstrations obtained in the real world.

A Geomagic Touch, supplied by 3D Systems, was used as a robot. This robot has six joints; three are active joints and the rest are passive joints. The passive joints were fixed to regard the robot as having three degrees of freedom.

5.3.2 Control System Setup

For collecting human demonstrations, a four-channel bilateral control system as illustrated in Fig. 5.6 was used. The use of bilateral control system allows a training dataset to include the constraints of the robots such as motion ranges, maximum torques, joint stiffness [78, 79]. This method has the following advantages:

- The same control system can be used in both the data collection phase and the production phase; thus, the dataset includes the dynamics of the robot and the controller.
- Because the bilateral control system can present realistic haptics to operators, human experts can operate the robot more intuitively compared with joysticks; as a result, the quality of demonstrations can be improved.
- Force control is included, allowing robots to adapt to small fluctuations of the environment at the controller level.

In the control system, $(K_\mathrm{p} + \frac{sg}{s+g}K_\mathrm{d})$ and K_f are a position controller and force controller, respectively. The control parameters are described in Table 5.1. A human expert grasps the master-side and tele-operates the slave-side through the bilateral control system. Both the master-side and slave-side were Geomagic Touch, with disturbance observers [49] and reaction torque observers [80] to estimate the external torques, $\boldsymbol{\tau}_\mathrm{m}^\mathrm{res}$ and $\boldsymbol{\tau}_\mathrm{s}^\mathrm{res}$. While the human expert is operating the robots, the movements are recorded: the master-side joint angle $\boldsymbol{\theta}_\mathrm{m}^\mathrm{res}(t)$, joint angular velocity $\dot{\boldsymbol{\theta}}_\mathrm{m}^\mathrm{res}(t)$, external torque $\boldsymbol{\tau}_\mathrm{m}^\mathrm{res}(t)$, slave-side joint angle $\boldsymbol{\theta}_\mathrm{s}^\mathrm{res}(t)$, joint angular velocity $\dot{\boldsymbol{\theta}}_\mathrm{s}^\mathrm{res}(t)$, and external torque $\boldsymbol{\tau}_\mathrm{s}^\mathrm{res}(t)$.

After the neural networks were trained, the master-side was replaced by the trained neural networks, as illustrated in Fig. 5.7. The orange parts in the control system were activated every $\Delta t_\mathrm{NN} = 20$ ms, whereas the rest of the control system was activated every $\Delta t = 1$ ms.

5.3.3 Training-Data Collection

Pushing motions were collected by placing the block at six patterns of positions, as shown in Fig. 5.5: left/right and 4.5/8.0/11.5 cm from the robot. Two kinds of dataset

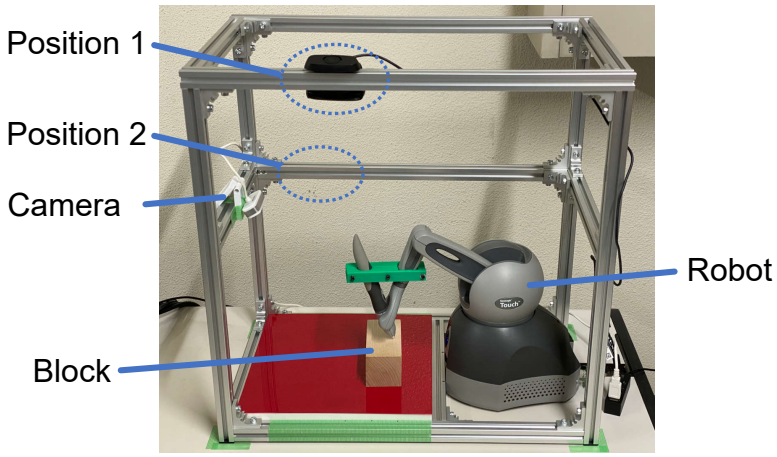


Figure 5.3: Task overview. Positions 1 and 2 indicate camera positions used in Section 5.3.5.

were collected: 18 pairs of images and motions; and 90 motions without images. The former image–motion dataset were used to train the CNN; whereas the latter motion-only dataset were used to train the SeqAE.

Here, the training dataset with imbalance numbers of images and motions simulate the case when the visual conditions changed. In a completely new visual condition such as different camera positions, old images in an original visual condition are useless, whereas motions can be used for training.

5.3.4 Model Setup

Figure 5.8 illustrates the model architectures. A CNN and a SeqAE were used. The CNN consisted of five convolutional layers (Conv) and a linear layer (Linear); each convolutional layer is followed by a ReLU activation and a BatchNormalization layer [81]. The SeqAE consisted of an encoder and decoder with seven layers of LSTMs and a linear layer. The latent representations $\boldsymbol{z}$ were set to eight dimensional.

Input/output variables of the SeqAE was designed according to Section 5.2.1. The encoder receives time series of the master and slave states, $\boldsymbol{\theta}_{\mathrm{m}}^{\mathrm{res}}(t)$, $\dot{\boldsymbol{\theta}}_{\mathrm{m}}^{\mathrm{res}}(t)$, $\boldsymbol{\tau}_{\mathrm{m}}^{\mathrm{res}}(t)$, $\boldsymbol{\theta}_{\mathrm{s}}^{\mathrm{res}}(t)$, $\dot{\boldsymbol{\theta}}_{\mathrm{s}}^{\mathrm{res}}(t)$, and $\boldsymbol{\tau}_{\mathrm{s}}^{\mathrm{res}}(t)$, and finally generates a latent representation $\boldsymbol{z}$; the decoder receives $\boldsymbol{z}$ and the current state of the slave-side, $\boldsymbol{\theta}_{\mathrm{s}}^{\mathrm{res}}(t)$, $\dot{\boldsymbol{\theta}}_{\mathrm{s}}^{\mathrm{res}}(t)$, and $\boldsymbol{\tau}_{\mathrm{s}}^{\mathrm{res}}(t)$, and generates the next control command for the slave-side, $\hat{\boldsymbol{\theta}}_{\mathrm{m}}^{\mathrm{res}}(t+\Delta t_{\mathrm{NN}})$, $\hat{\dot{\boldsymbol{\theta}}}_{\mathrm{m}}^{\mathrm{res}}(t+\Delta t_{\mathrm{NN}})$, and $\hat{\boldsymbol{\tau}}_{\mathrm{m}}^{\mathrm{res}}(t+\Delta t_{\mathrm{NN}})$.

The SeqAE and CNN were trained in 2000 and 100 epochs, respectively. The mini-batch size was set to 16 and eight for the SeqAE and CNN, respectively. Adam [47] was used for optimization.

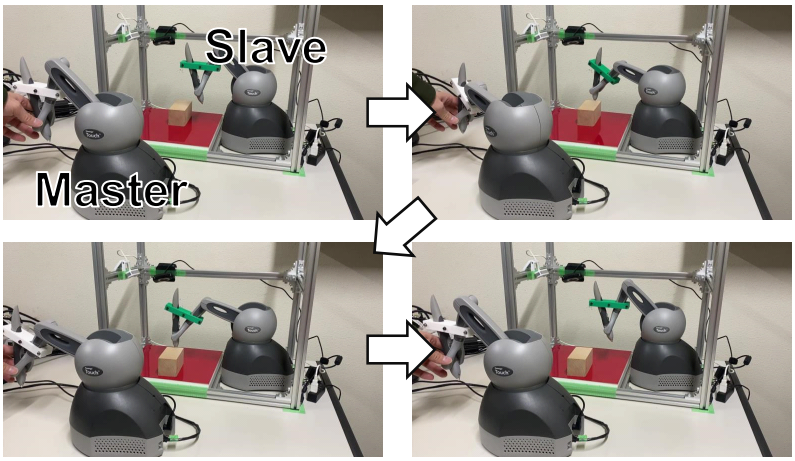


Figure 5.4: Snapshots of a demonstration

Figure 5.5: Examples of training images

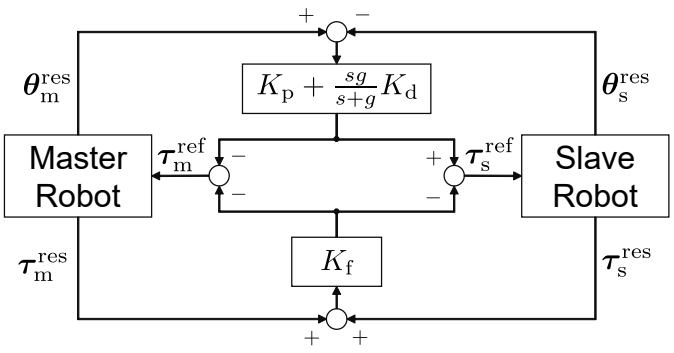


Figure 5.6: Bilateral control system

5.3.5 Results

Comparison with end-to-end learning

The proposed method was compared with end-to-end learning. Two models with the same architecture were trained—one was trained by the proposed method with 90 motions and 18 images, and the other was trained by end-to-end learning with 18 image-motion pairs. The end-to-end learning model was trained with 2000 epochs. It must be noted

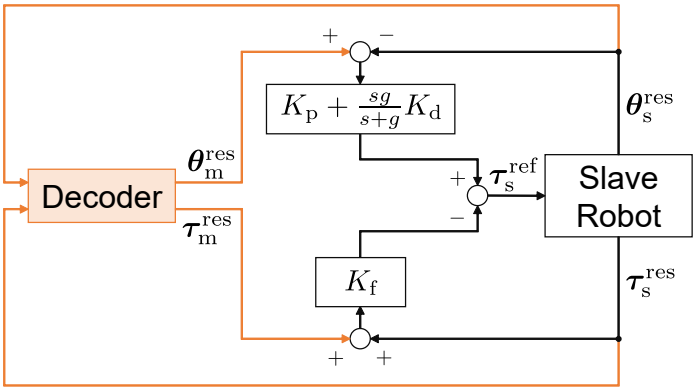


Figure 5.7: Control system when the decoder is used

Table 5.1: Control parameters

Item	Value
Position control gain K_p	120
Velocity control gain K_d	10
Force control gain K_f	0.5
Cutoff frequency of filters g [Hz]	10
Control interval Δt [s]	0.001

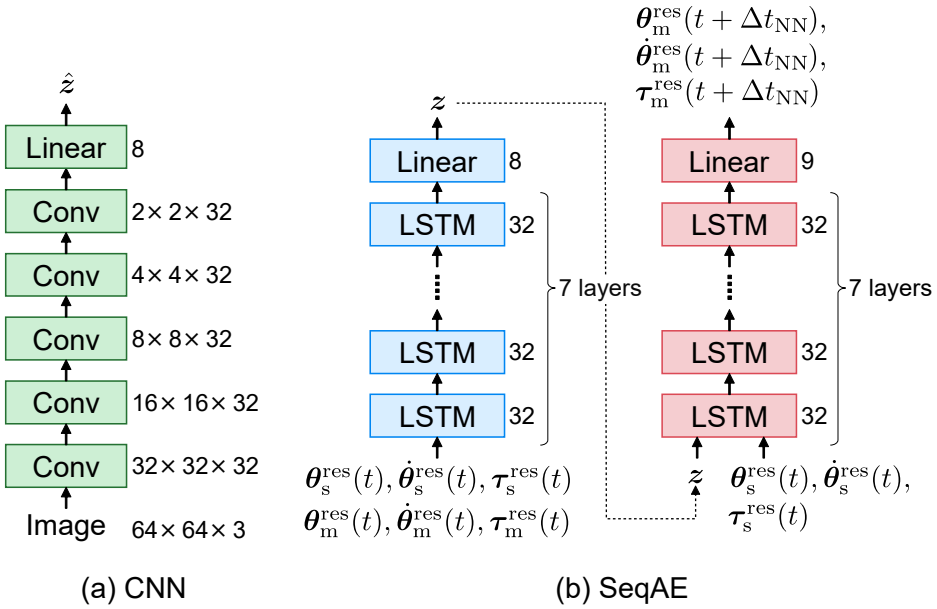


Figure 5.8: Model architecture

that the same number of image-motion pairs were used for both models.

The performances of these trained models in the pushing task were evaluated at six positions as described in Section 5.3.3. Figure 5.9 presents the success rates in 30 trials

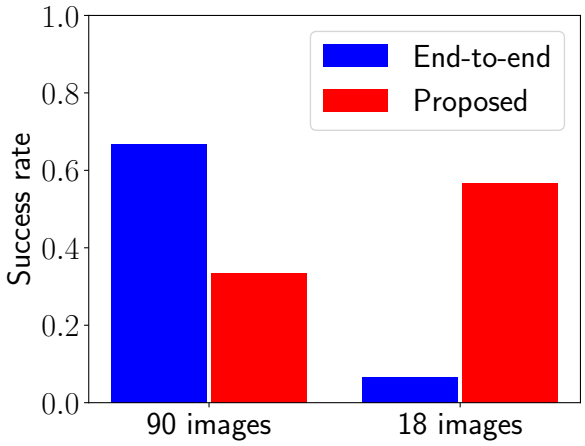
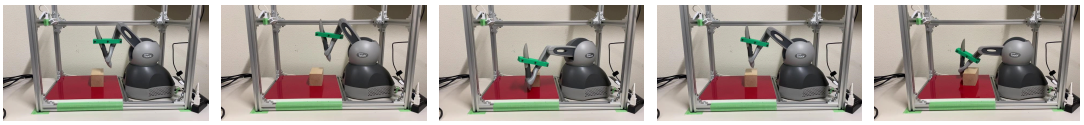


Figure 5.9: Success rates of 30 trials. In end-to-end learning, the success rate decreased when the number of images was small. In the proposed method, the success rate did not decrease even when the number of images was small.



(a) End-to-end learning: the robot significantly shook the arm and finally failed in the task.

(b) Proposed method: the robot succeeded in performing the task.

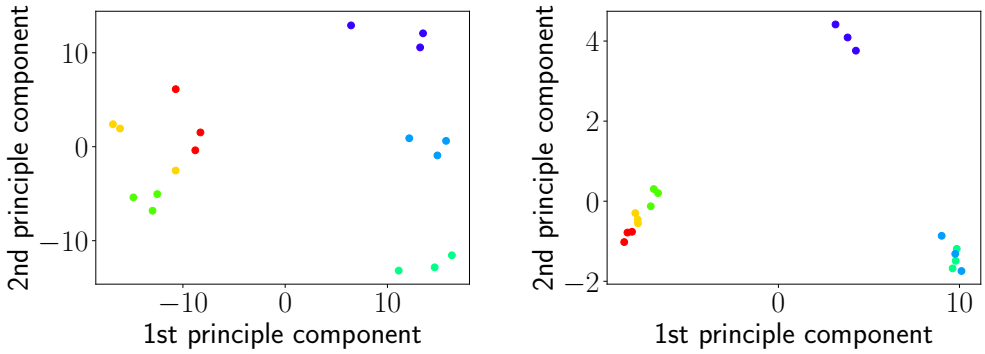
Figure 5.10: Snapshots

(*i.e.,* five times at each of the six positions). The success condition has been described in Section 5.3.1. In the figure, the same models trained with 90 image-motion pairs were also evaluated. When the number of images was large, end-to-end learning delivered a success rate of more than 60%; however, the success rate significantly decreased when the number of images was small. In contrast, the success rate of the proposed method did not decrease even when the number of images was small. Figure 5.10 shows snapshots of the generated motions. In case of end-to-end learning, the model often generated confused motions, where the arm exhibited a significant shaking motion. In the proposed method, the model generated stable and valid motions, similar to the demonstrations.

To exclude the possibility that these results were caused by the specific model architecture, two additional models with different numbers of layers and units from those of the model described in Section 5.3.4 (named *original*) were evaluated. One had an encoder/decoder RNN with 10 layers with 128 LSTMs (named *larger*), and the other one had an encoder/decoder with 3 layers with 32 LSTMs (named *smaller*). Table 5.2 presents the success rates of these models. Although there were slight differences, the same ten-

Table 5.2: Comparison of the success rate with different models

Training method	Original	Larger	Smaller
The proposed method	**56.7%**	**30.0%**	**46.7%**
End-to-end learning	6.7%	6.7%	33.3%



(a) End-to-end learning

(b) Proposed method

Figure 5.11: Visualization of the latent representations of images. The same color indicates the same block position. In both cases, the latent representations of images were clustered.

dency as in Fig. 5.9 was confirmed in the relationship between the proposed method and the end-to-end learning, even in different models. Therefore, it can be concluded that the use of SeqAE improves the task performance compared to end-to-end learning when the number of images is smaller than the number of motions.

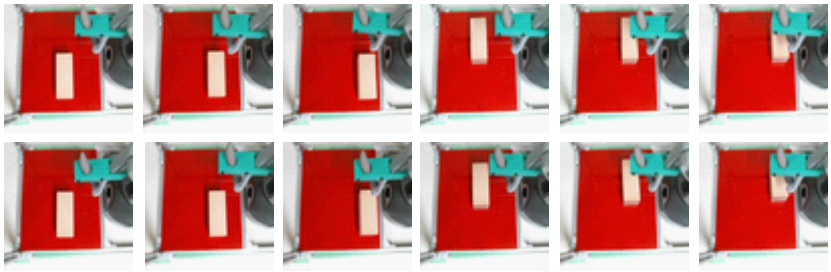
Visualization of latent spaces

To discuss whether these results are primarily caused by the CNN or the RNN side, the latent representations of images of the trained models were visualized. If the performance difference between the proposed method and end-to-end learning was caused by insufficient training of CNNs, the latent representations of images are expected to not express the block positions; otherwise, it can be said that the CNNs could distinguish the block positions.

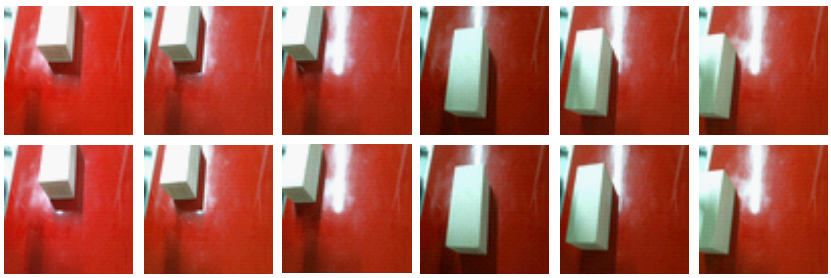
Figure 5.11 shows the latent spaces projected into two-dimensional spaces by the principal component analysis. Here, latent representations of 18 images not used for training were visualized. In both end-to-end learning and the proposed method, the latent representations constructed clusters. The results indicate that both methods succeeded in extracting the image features. Therefore, it can be concluded that the poor performance of end-to-end learning was caused by the RNN side; it reinforces the effectiveness of the use of SeqAE in improving the robustness of the RNN side.

Changes in visual conditions

Finally, the proposed method in the case where the camera positions changed was demonstrated as a typical example of changes in visual conditions. Here, two different



(a) Position 1



(b) Position 2

Figure 5.12: Examples of images captured at different positions

Table 5.3: Comparison of the success rate with different camera positions.

Camera position	Success rate
Default position	56.7%
Position 1	40.0%
Position 2	26.7%

camera positions were used as shown in Fig. 5.3; examples of images are shown in Fig. 5.12. Because the appearances of the block in the images were significantly different, the same set of images could no longer be used for training. 18 images with motions for each camera position were collected to train a CNN with trained SeqAE.

Table 5.3 shows the success rates for the different camera positions in 30 trials. Although there were variations in the results, the success rates in all these cases were higher than that achieved by end-to-end learning, as detailed in Table 5.2; even though the success rate in Position 2 was lower than the others, it was still higher than end-to-end learning with the same model architecture. Therefore, it can be concluded that SeqAEs can be reused when visual conditions changed.

5.4 Conclusion

LfD with image inputs (that correspond to external orders) has some issues: 1) frequent changes in visual conditions and 2) instability of motion generation caused by teacher forcing. To solve these issues, a method of using latent representations of SeqAEs was proposed. By training SeqAEs, a motion generator that is robust against state-variables

fluctuations could be obtained. Moreover, SeqAEs can be reused even when the visual conditions change. It was confirmed that the neural models trained by the proposed method could generate stable and valid motions even when the number of images decreased. Also, the proposed method in the case where the camera position changed was demonstrated. These results demonstrate the effectiveness of the proposed motion generators that use latent representations.

Chapter 6

Conclusion

Although robots are expected to extend their applications to our daily tasks, the current robots are not suitable for these tasks as the tasks are executed in diverse situations. In this study, I focused on the diversity of external orders to motion generators. This can be realized by machine learning methods that associate trajectories with certain intermediate representation as opposed to using the external orders directly. The use of the intermediate representation resolves the issue of the diversity of representations in the external orders. Here, the intermediate representation is to be associated with the external orders after training the motion generators. Considering these intermediate representations, I aimed to use the latent representations of trajectories, particularly, seq2seq models that can extract latent representations from trajectories and generate motions online. Although seq2seq models are considered to be an efficient approach for addressing the realization of motion generators that can handle various external orders, several topics are to be firstly understood. Therefore, these topics were discussed in this dissertation.

First, training methods of various trajectories were researched. Although the use of the latent representations enables training independent of certain external orders, the variety of training trajectories limits the performance of the models. Therefore, trajectory deformation-based training methods for seq2seq models were proposed. The models are trained to deform given trajectories to satisfy the constraints and dynamics. Finally, users can train various valid trajectories to the models and control the diversity of trajectories.

Second, the association of certain external orders with latent representations was researched. As the seq2seq models do not specify what the latent representations represent, they have low interpretability. In this study, association methods of the latent representations with two types of the external order were proposed—mathematical expressions and numerical expressions. For mathematical expressions, a backpropagation-based optimization method for the latent representations was proposed. Using the proposed method, the latent representations can be optimized to minimize the given objective functions more efficiently than the direct optimization of trajectories. In addition, for numerical expressions, such as demonstration data, a training method for latent representations was proposed.

Further, the robustness of the trained decoders was evaluated. They demonstrated robustness against fluctuations in environmental parameters and exposure bias.

In summary, using seq2seq models, various trajectories can be trained solely with simplified environment models and constraints. Furthermore, the latent representations can be associated with various objective functions and numerical data after training. Moreover, the trained decoders, which are motion generators that can be used online, are robust against exposure bias and fluctuations in environmental parameters. By integrating the proposed methods, motion generators that can handle various external orders can be realized based on seq2seq models.

Bibliography

[1] T. Tsuji, K. Kutsuzawa, and S. Sakaino, "Optimized Trajectory Generation based on Model Predictive Control for Turning Over Pancakes," *IEEJ J. Ind. Appl.*, vol. 7, no. 1, pp. 22–28, jan 2018.

[2] K. M. Lynch and M. T. Mason, "Dynamic underactuated nonprehensile manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 2, 1996, pp. 889–896.

[3] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 4066–4073.

[4] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, may 2001.

[5] N. Hirose, R. Tajima, and K. Sukigara, "MPC policy learning using DNN for human following control without collision," *Adv. Robot.*, vol. 32, no. 3, pp. 148–159, 2018.

[6] M. Inoue, T. Yamashita, and T. Nishida, "Robot Path Planning by LSTM Network Under Changing Environment," *Proc. Int. Conf. Comput. Sci. Adv. Database Comput.*, 2017.

[7] S. R. Carvalho, R. Boulic, C. A. Vidal, and D. Thalmann, "Latent motion spaces for full-body motion editing," *Vis. Comput.*, vol. 29, no. 3, pp. 171–188, 2013.

[8] H. Yin, F. S. Melo, B. Aude, and A. Paiva, "Associate Latent Encodings in Learning from Demonstrations," *Proc. Thirty-First AAAI Conf. Artif. Intell.*, vol. 31, no. 641, pp. 3848–3854, 2017.

[9] D. Takahashi and S. Katsura, "Extended Reproduction of Demonstration Motion Using Variational Autoencoder," in *Proc. IEEE Int. Symp. Ind. Electron.*, 2018, pp. 1057–1062.

[10] B. Ichter and M. Pavone, "Robot Motion Planning in Learned Latent Spaces," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2407–2414, feb 2019.

[11] J. D. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine, "Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Tra-

jectory Embeddings," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, jun 2018, pp. 1009–1018.

[12] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an Embedding Space for Transferable Robot Skills," pp. 1–16, 2018.

[13] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning Latent Plans from Play," *arXiv Prepr. arXiv1903.01973*, mar 2019.

[14] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2014, pp. 1724–1734.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[16] O. Vinyals and Q. Le, "A Neural Conversational Model," *arXiv Prepr. arXiv1506.05869*, jun 2015. [Online]. Available: https://arxiv.org/abs/1506.05869

[17] R. Nallapati, B. Zhou, C. N. dos Santos, Ç. Gülçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proc. SIGNLL Conf. Comput. Nat. Lang. Learn.*, 2016, pp. 280–290.

[18] T. Yamada, S. Murata, H. Arie, and T. Ogata, "Representation Learning of Logic Words by an RNN: From Word Sequences to Robot Actions," *Front. Neurorobot.*, vol. 11, p. 70, 2017.

[19] K. Kutsuzawa, S. Sakaino, and T. Tsuji, "Learning identity mapping of trajectories by sequence-to-sequence model with time series chunking," in *Proc. IEEJ Int. Work. Sensing, Actuation, Motion Control. Optim.*, 2017, pp. 1–6.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] F. Gers, "Long short-term memory in recurrent neural networks," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, 2001.

[22] K. Kutsuzawa, "Realization of sequence-to-sequence models that deform trajectories for dynamic manipulation," Master's thesis, 2017, in Japanese.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[24] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Adv. Neural Inf. Process. Syst. 28*, 2015, pp. 1171–1179.

[25] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence Level Training with Recurrent Neural Networks," *arXiv Prepr. arXiv1511.06732*, nov 2015.

[26] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[27] K. M. Lynch and M. T. Mason, "Dynamic Nonprehensile Manipulation: Controllability, Planning, and Experiments," *Int. J. Rob. Res.*, vol. 18, no. 1, pp. 64–92, 1999.

[28] K. M. Lynch, N. Shiroma, H. Arai, and K. Tanie, "The roles of shape and motion in dynamic manipulation: the butterfly example," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, 1998, pp. 1958–1963.

[29] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2010, pp. 3232–3237.

[30] M. Bauza and A. Rodriguez, "A probabilistic data-driven model for planar pushing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3008–3015.

[31] K. M. Lynch, "Nonprehensile Robotic Manipulation : Controllability and Planning," Ph.D. dissertation, Carnegie Mellon University, 1996.

[32] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to Poke by Poking: Experiential Learning of Intuitive Physics," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 5074–5082.

[33] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-To-End Learning from Demonstration," in *Proc. 2018 IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3758–3765.

[34] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," *arXiv Prepr. arXiv1710.06537*, oct 2017.

[35] M. T. Mason and K. M. Lynch, "Dynamic manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 1, 1993, pp. 152–159.

[36] T. Tsuji, J. Ohkuma, and S. Sakaino, "Dynamic object manipulation considering contact condition of robot with tool," *IEEE Trans. Ind. Electron.*, vol. 63, no. 3, pp. 1972–1980, 2016.

[37] P. Lertkultanon and Q. C. Pham, "Dynamic Non-prehensile Object Transportation," in *Proc. 2014 13th Int. Conf. Control Autom. Robot. Vis.*, 2014, pp. 1392–1397.

[38] H. Kurniawati and T. Fraichard, "From path to trajectory deformation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2007, pp. 159–164.

[39] F. Lamiraux and D. Bonnafous, "Reactive trajectory deformation for nonholonomic systems: application to mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, 2002, pp. 3099–3104.

[40] P. Zegers and M. K. Sundareshan, "Trajectory Generation and Modulation Using Dynamic Neural Networks," *IEEE Trans. Neural Networks*, vol. 14, no. 3, pp. 520–533, 2003.

[41] T. Nierhoff and S. Hirche, "Fast trajectory replanning using Laplacian mesh optimization," in *Proc. Int. Conf. Control. Autom. Robot. Vis.*, 2012, pp. 154–159.

[42] A. Pekarovskiy, T. Nierhoff, J. Schenek, Y. Nakamura, S. Hirche, and M. Buss, "Online deformation of optimal trajectories for constrained nonprehensile manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2015, pp. 2481–2487.

[43] S. Murata, Y. Yamashita, H. Arie, T. Ogata, S. Sugano, and J. Tani, "Learning to perceive the world as probabilistic or deterministic via interaction with others: a neuro-robotics experiment," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 4, pp. 830–848, 2015.

[44] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[45] A. M. Dai and Q. V. Le, "semi-supervised sequence learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3079–3087.

[46] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv Prepr. arXiv1607.06450*, 2016.

[47] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–13.

[48] K. Ohishi, K. Ohnishi, and K. Miyachi, "Torque - Speed Regulation of Dc Motor Based on Load Torque Estimation Method." in *Proc. JIEE/1983 Int. Power Electron. Conf.*, vol. 2, 1983, pp. 1209–1218.

[49] K. Ohnishi, M. Shibata, and T. Murakami, "Motion control for advanced mechatronics," *IEEE/ASME Trans. Mechatronics*, vol. 1, no. 1, pp. 56–67, 1996.

[50] M. Higashimori, K. Utsumi, Y. Omoto, and M. Kaneko, "Dynamic Manipulation Inspired by the Handling of a Pizza Peel," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 829–838, 2009.

[51] T. H. Vose, P. Umbanhowar, and K. M. Lynch, "Sliding manipulation of rigid bodies on a controlled 6-DoF plate," *Int. J. Rob. Res.*, vol. 31, no. 7, pp. 819–838, 2012.

[52] J. Shi, J. Z. Woodruff, and K. M. Lynch, "Dynamic in-hand sliding manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2015, pp. 870–877.

[53] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic Motion Planning," *J. ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[54] S. Levine, N. Wagener, and P. Abbeel, "Learning Contact-Rich Manipulation Skills with Guided Policy Search," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 156–163.

[55] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang, "Rearrangement with Nonprehensile Manipulation Using Deep Reinforcement Learning," in *2018 IEEE Int. Conf. Robot. Autom.*, mar 2018, pp. 270–277.

[56] I. Mordatch, K. Lowrey, G. Andrew, Z. Popovic, and E. Todorov, "Interactive Control of Diverse Complex Characters with Neural Networks," in *Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–8.

[57] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search," *2016 IEEE Int. Conf. Robot. Autom.*, pp. 528–535, 2016.

[58] D. Furuta, K. Kutsuzawa, T. Okamoto, S. Sakaino, and T. Tsuji, "Model Predictive Control based Deep Neural Network for Dynamic Manipulation," in *Proc. Annu. Conf. IEEE Ind. Electron. Soc.*, 2017, pp. 5215–5220.

[59] A. Nguyen, J. Yosinski, and J. Clune, "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, dec 2015, pp. 427–436.

[60] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Adv. Neural Inf. Process. Syst. 29*, 2016, pp. 1–9.

[61] R. Pahic, Z. Loncarevic, A. Ude, B. Nemec, and A. Gams, "User Feedback in Latent Space Robotic Skill Learning," *IEEE-RAS Int. Conf. Humanoid Robot.*, pp. 791–797, 2018.

[62] A. Byravan, F. Lceb, F. Meier, and D. Fox, "SE3-Pose-Nets: Structured deep dynamics models for visuomotor control," in *Proc. 2018 IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3339–3346.

[63] K. Kawaharazuka, T. Ogawa, J. Tamura, and C. Nabeshima, "Dynamic Manipulation of Flexible Objects with Torque Sequence Using a Deep Neural Network," in *Proc. 2019 Int. Conf. Robot. Autom.*, 2019, pp. 2139–2145.

[64] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," *arXiv Prepr. arXiv1511.06349*, nov 2015.

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

[66] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: a survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 21:1–21:35, 2017.

[67] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *Proc. Int. Conf. Mach. Learn.*, vol. 97, 1997, pp. 12–20.

[68] E. Talvitie, "Model regularization for stable sample rollouts," in *Proc. 30th Conf. Uncertain. Artif. Intell.*, 2014, pp. 780–789.

[69] H. Zhang, E. Heiden, S. Nikolaidis, J. J. Lim, and G. S. Sukhatme, "Auto-conditioned Recurrent Mixture Density Networks for Learning Generalizable Robot Skills," *arXiv Prepr. arXiv1810.00146*, sep 2018.

[70] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-Supervised Learning with Deep Generative Models," in *Adv. Neural Inf. Process. Syst. 27*, 2014, pp. 3581–3589.

[71] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational Autoencoder for Deep Learning of Images, Labels and Captions," in *Adv. Neural Inf. Process. Syst. 29*, 2016, pp. 2352–2360.

[72] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 397–403, apr 2017.

[73] T. Inoue, S. Choudhury, G. De Magistris, and S. Dasgupta, "Transfer Learning from Synthetic to Real Images Using Variational Autoencoders for Precise Position Detection," in *Proc. 2018 25th IEEE Int. Conf. Image Process.*, 2018, pp. 2725–2729.

[74] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep Predictive Policy Training using Reinforcement Learning," in *Proc. 2017 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2017, pp. 2351–2358.

[75] A. Hämäläinen, K. Arndt, A. Ghadirzadeh, and V. Kyrki, "Affordance Learning for End-to-End Visuomotor Robot Control," *arXiv Prepr. arXiv1903.04053*, mar 2019.

[76] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task Sequence to Sequence Learning," *arXiv Prepr.*, vol. arXiv: 151, nov 2015.

[77] P. Ramachandran, P. J. Liu, and Q. V. Le, "Unsupervised Pretraining for Sequence to Sequence Learning," *arXiv Prepr. arXiv1611.02683*, nov 2016.

[78] T. Adachi, K. Fujimoto, S. Sakaino, and T. Tsuji, "Imitation Learning for Object Manipulation Based on Position/Force Information Using Bilateral Control," in *Proc. 2018 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2018, pp. 3648–3653.

[79] K. Fujimoto, S. Sakaino, and T. Tsuji, "Time Series Motion Generation Considering Long Short-Term Motion," in *Proc. 2019 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2019, pp. 6842–6848.

[80] T. Murakami, F. Yu, and K. Ohnishi, "Torque sensorless control in multidegree-of-freedom manipulator," *IEEE Trans. Ind. Electron.*, vol. 40, no. 2, pp. 259–265, 1993.

[81] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv Prepr. arXiv1502.03167*, pp. 1–11, 2015.

Acknowledgements

This dissertation summarizes my research conducted in the Tsuji laboratory. Since April 2014, I received a lot of help from many people for my research. I would like to express my acknowledgement to them.

First of all, I am grateful Associate Professor Dr. Toshiaki Tsuji at Saitama University for his support and tolerance. Although I was not good at presenting my research straightforwardly, he patiently guided me in preparing papers and presentations that have a logical and simple structure. He also taught and demonstrated to me what great researchers are. His research perspective has created interest in me to become a researcher. I could widen my view thanks to him giving me many chances to attend conferences and meet other researchers.

I am also grateful to the members of my Ph.D. dissertation committee, Professor Dr. Yoshinori Kuno, Professor Dr. Yasuyoshi Kaneko, and Professor Dr. Hiroyuki Yaguchi, at Saitama University. Their comments and advice helped me to improve this dissertation.

I am also grateful to Associate Professor Dr. Sho Sakaino at the University of Tsukuba for the fruitful discussions with him. From when he was at Saitama University, he gave me many severe yet essential comments. Discussions with him built my mental strength.

Assistant Professor Dr. Takahiro Nozaki at Keio University taught me about the preparedness required for a doctoral student. He was patient during my consultations with him about my worries regarding the doctoral program. He also gave me opportunities to meet many researchers and doctoral students.

I would like to thank the Japan Society for the Promotion of Science (JSPS), which supported me financially.

I am thankful to the members in Tsuji laboratory and Sakaino laboratory. Especially, Mr. Tomoya Kitamura, my colleague, helped me a lot during my research life in the doctoral program. He often gave me many helpful comments on my research. Moreover, his thought of always not forgetting fun and playfulness has helped me mentally.

Finally, I am grateful to my parents—my father, who passed away when I was 17 years old, and my mother, who raised me and supported me throughout the doctoral program.

There are many people I could not mention here. I express my sincere gratitude to everybody who have supported me.

March, 2020
Kyo Kutsuzawa