Sequence-to-Sequence Model for Trajectory Planning of Nonprehensile Manipulation Including Contact Model

Kyo Kutsuzawa, Sho Sakaino, and Toshiaki Tsuji

***Abstract*—Nonprehensile manipulation is necessary for robots to operate in humans' daily lives. As nonprehensile manipulation should satisfy both kinematics and dynamics requirements simultaneously, it is difficult to manipulate objects along given paths. Previous studies have considered the problems with sequence-to-sequence models, which are neural networks for time-series conversion. However, they did not consider nonlinear contact models, such as friction models. When we train the seq2seq models using end-to-end backpropagation, training losses vanish owing to static friction. In this letter, we realize sequence-to-sequence models for trajectory planning of nonprehensile manipulation including contact models between the robots and target objects. This letter proposes a training curriculum that commences training without contact models to bring the seq2seq models outside of the gradient-vanishing zone. This letter discusses sliding manipulation, which includes a friction model between objects and tools, such as frying pans fixed onto the robots. We validated the proposed curriculum through a simulation. In addition, we observed that the trained seq2seq models could handle parameter fluctuations that did not exist during training.**

***Index Terms*—Deep learning in robotics and automation, motion and path planning, nonprehensile manipulation, sequence-to-sequence model.**

I. Introduction

ROBOTS are currently primarily used for industrial purposes; however, they are also expected to function in humans' daily lives. The target of automation by robots will encompass a wide range of tasks such as cooking meals, cleaning rooms, and carrying luggage.

To accomplish these tasks, object manipulation is important. Cooking robots need to manipulate foods with cooking tools and cleaning robots need to put furniture aside while cleaning. While object manipulation involves grasping in many cases, a type of object manipulation that does not involve grasping is also important, especially in an open environment. Lynch *et al.* called such type of manipulation as nonprehensile manipulation [1]. For example, robots can utilize inertial force to manipulate objects on a frying pan without dropping even when the frying pan tilts largely [2]–[4].

As nonprehensile manipulation should satisfy both dynamics and kinematics requirements simultaneously, motion planning is a difficult issue. Dynamic programming is a solution to the problem [5], [6]. Pekarovskiy *et al.* realized a planar volleyball task by trajectory deformation using a least-squares approach [7], [8]. Tsuji *et al.* realized a trajectory planning method for turning over pancakes with a spatula [4]. As the control system of nonprehensile manipulation is diverse and complicated to design by hand, machine learning is an effective approach. Kormushev *et al.* realized a robot flipping pancakes by reinforcement learning [9]. Bauza *et al.* realized a planar pushing motion using a variation in the Gaussian process [10]. Neural networks have attracted attention in some studies recently [11]–[13].

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, Kutsuzawa *et al.* proposed motion planning methods with sequence-to-sequence (seq2seq) models [14], [15], which are neural networks applicable for time series conversion [16], [17]. They realized a robot turning over a pancake with a spatula using seq2seq models that deform trajectories to avoid dropping the pancake. 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, we should extend the seq2seq models to cases where complex dynamics exist, such as the contact models between the robots and the objects.

This letter discuss the sliding manipulation [18]–[20], 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 [21].

Manuscript received February 23, 2018; accepted June 17, 2018. Date of publication July 11, 2018; date of current version August 2, 2018. This letter was recommended for publication by Associate Editor N. Dantam and Editor T. Asfour upon evaluation of the reviewers' comments. This work was supported by the JSPS Grant-in-Aid for Scientific Research (B), no. 16H04311. *(Corresponding author: Kyo Kutsuzawa.)*

K. Kutsuzawa and T. Tsuji are with the Graduate School of Science and Engineering, Saitama University, Saitama 338-8570, Japan (e-mail: k.kutsuzawa.430@ms.saitama-u.ac.jp; tsuji@ees.saitama-u.ac.jp).

S. Sakaino is with the JST PRESTO and the Graduate School of Science and Engineering, Saitama University, Saitama 338-8570, Japan (e-mail: sakaino@mail.saitama-u.ac.jp).

Digital Object Identifier 10.1109/LRA.2018.2854958

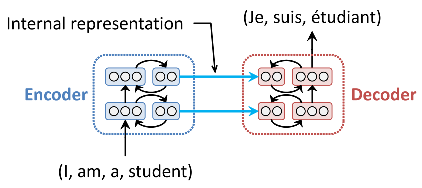


Fig. 1. Sequence-to-sequence model for machine translation.

In this letter, we aim to handle the issue by applying curriculum learning [22]. This letter proposes a training curriculum that does not use the contact model at the beginning of the training. The contributions of this paper are as follows:

1) This letter proposes seq2seq models for motion planning considering nonlinear contact models. This letter 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 letter 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 letter shows that the seq2seq models trained by the proposed curriculum can handle fluctuations of friction parameters even without special training.

The remainder of this paper consists of the following sections. Section II explains the proposed neural model and the physics model. Section III explains the proposed training method based on curriculum learning. Section IV describes the concrete implementation of the proposed method. Section V shows the validity of the proposed method by simulation. Finally, Section VI presents the conclusion and future works.

II. Sequence-to-Sequence Model for Sliding Manipulation

A. Sequence-to-Sequence Model

Sequence-to-sequence (seq2seq) models [14], [15] are neural network architectures for converting a sequence to another sequence. Seq2seq models are primarily applied to natural language processing tasks such as machine translation [14], [15], conversational modeling [23], and text summarizing [24].

A basic seq2seq model consists of two recurrent neural networks (RNNs): an encoder and a decoder. An example of the conversion process in machine translation is illustrated in Fig. 1. First, the encoder receives a source sentence, word by word. The encoder converts the received words to an internal representation and memorizes them into internal memories. After the encoding process, the encoder's internal representation reflects the meaning of the sentence [15]. Subsequently, the encoder initializes the decoder with the encoder's internal representation. The decoder receives a special token representing the start of sentence and generates the first word of the translated sentence according to the internal representation. Subsequently, the decoder receives the generated word, generates the next word, and repeats the process. Finally, the decoding process finishes when the decoder generates a special token representing the end of sentence.

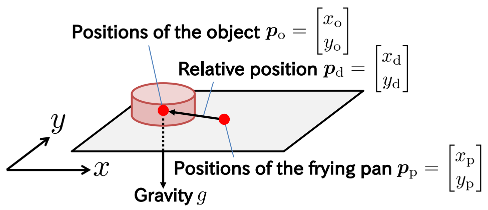

Positions of the object $p_\mathrm{o} = \begin{bmatrix} x_\mathrm{o} \\ y_\mathrm{o} \end{bmatrix}$
Relative position $p_\mathrm{d} = \begin{bmatrix} x_\mathrm{d} \\ y_\mathrm{d} \end{bmatrix}$
y
x
Positions of the frying pan $p_\mathrm{p} = \begin{bmatrix} x_\mathrm{p} \\ y_\mathrm{p} \end{bmatrix}$
Gravity g

Fig. 2. Overview of the physics model.

Recently, some studies applied the seq2seq models to robotics. Yamada *et al.* applied the seq2seq models to robots to learn the relationship between linguistic commands and appropriate actions [25]. Kutsuzawa *et al.* proposed a method of dividing a trajectory into groups of samples to solve an issue caused by the seq2seq models processing a long time-series of robot motions [16], [17].

B. Physics Model

This study uses two-dimensional physics and is a typical example that includes nonlinear contact models. Fig. 2 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

$$\boldsymbol{p}_\mathrm{d}[k] \equiv \boldsymbol{p}_\mathrm{o}[k] - \boldsymbol{p}_\mathrm{p}[k] \tag{1}$$

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:

$$\boldsymbol{x}_\mathrm{p}[k] \equiv \left[\boldsymbol{p}_\mathrm{p}^\top[k], \dot{\boldsymbol{p}}_\mathrm{p}^\top[k]\right]^\top = \left[x_\mathrm{p}[k], y_\mathrm{p}[k], \dot{x}_\mathrm{p}[k], \dot{y}_\mathrm{p}[k]\right]^\top. \tag{2}$$

Similarly, we define $\boldsymbol{x}_\mathrm{o}[k]$ and $\boldsymbol{x}_\mathrm{d}[k]$.

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

$$\boldsymbol{x}_\mathrm{p}[k+1] = \boldsymbol{A}\boldsymbol{x}_\mathrm{p}[k] + \boldsymbol{B}\boldsymbol{u}_\mathrm{p}[k], \tag{3}$$

$$\boldsymbol{x}_\mathrm{d}[k+1] = \boldsymbol{A}\boldsymbol{x}_\mathrm{d}[k] + \boldsymbol{B}\boldsymbol{u}_\mathrm{d}[k], \tag{4}$$

where,

$$\boldsymbol{u}_\mathrm{p}[k] = \left[\ddot{x}_\mathrm{p}[k], \ddot{y}_\mathrm{p}[k]\right]^\top, \tag{5}$$

$$\boldsymbol{u}_\mathrm{d}[k] = \left[\ddot{x}_\mathrm{d}[k], \ddot{y}_\mathrm{d}[k]\right]^\top, \tag{6}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_2 & \Delta t\boldsymbol{I}_2 \\ \boldsymbol{O}_2 & \boldsymbol{I}_2 \end{bmatrix}, \tag{7}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{\Delta t^2}{2}\boldsymbol{I}_2 \\ \Delta t\boldsymbol{I}_2 \end{bmatrix}. \tag{8}$$

$\boldsymbol{u}_\mathrm{p}[k]$ 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

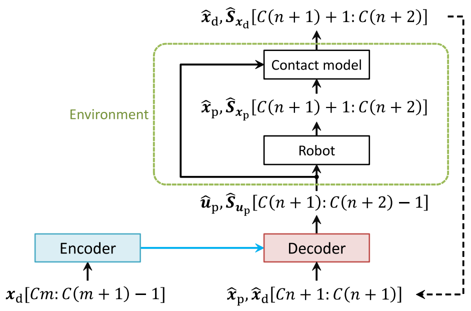

$\hat{x}_d, \hat{S}_{x_d}[C(n+1)+1: C(n+2)]$
Contact model
Environment
$\hat{x}_p, \hat{S}_{x_p}[C(n+1)+1: C(n+2)]$
Robot
$\hat{u}_p, \hat{S}_{u_p}[C(n+1): C(n+2)-1]$
Encoder
Decoder
$x_d[Cm: C(m+1)-1]$
$\hat{x}_p, \hat{x}_d[Cn+1: C(n+1)]$

Fig. 3. Architecture of the proposed seq2seq model.

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 [20]:

$$\boldsymbol{u}_\mathrm{d}[k] = \begin{cases} -\boldsymbol{u}_\mathrm{p}[k] - \mu' g \frac{\dot{\boldsymbol{p}}_\mathrm{d}[k]}{\|\dot{\boldsymbol{p}}_\mathrm{d}[k]\|} & \text{if } \dot{\boldsymbol{p}}_\mathrm{d}[k] \neq \boldsymbol{0} \\ -\boldsymbol{u}_\mathrm{p}[k] + \mu g \frac{\boldsymbol{u}_\mathrm{p}[k]}{\|\boldsymbol{u}_\mathrm{p}[k]\|} & \text{if } \dot{\boldsymbol{p}}_\mathrm{d}[k] = \boldsymbol{0} \text{ and } \|\boldsymbol{u}_\mathrm{p}[k]\| \geq \mu g \\ \boldsymbol{0} & \text{if } \dot{\boldsymbol{p}}_\mathrm{d} = \boldsymbol{0} \text{ and } \|\boldsymbol{u}_\mathrm{p}[k]\| < \mu g \end{cases} \cdot \tag{9}$$

Here, $g = 9.8\,\mathrm{m/s^2}$ 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.

C. Seq2seq Model Architecture

We used the seq2seq model as the motion-to-motion conversion, as proposed in [16], [17]. 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 [16]. This method can reduce the memory consumption of computers and enable stable training.

Fig. 3 illustrates the primary architecture of the proposed seq2seq model. For a given k_1 and k_2, let $\boldsymbol{x}_\mathrm{p}[k_1 : k_2]$ represent a sequence of state vectors of the frying pan from the k_1-th sample to the k_2-th sample as follows:

$$\boldsymbol{x}_\mathrm{p}[k_1 : k_2] = (\boldsymbol{x}_\mathrm{p}[k_1], \boldsymbol{x}_\mathrm{p}[k_1+1], \ldots, \boldsymbol{x}_\mathrm{p}[k_2]). \tag{10}$$

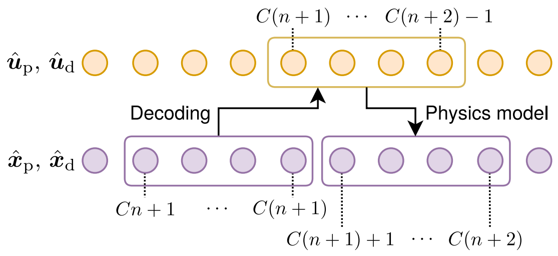


Fig. 4. Relationship between time series of $\hat{\boldsymbol{u}}_\mathrm{p}$, $\hat{\boldsymbol{x}}_\mathrm{p}$, $\hat{\boldsymbol{u}}_\mathrm{d}$, and $\hat{\boldsymbol{x}}_\mathrm{d}$ in the decoding phase. Each circle indicates a variable at a sample. Here illustrates the case of $C = 4$ as an example.

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

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:

$$\begin{aligned} &\boldsymbol{x}_\mathrm{d}[Cm : C(m+1)-1] \\ &\quad = (\boldsymbol{x}_\mathrm{d}[Cm], \ldots, \boldsymbol{x}_\mathrm{d}[C(m+1)-1]) . \end{aligned} \tag{11}$$

Here, m is the index of iterations of the encoding. C is the number of the samples the seq2seq models process at a time [16]. 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. 4. In each iteration, the decoder receives part of the time-series of state vectors as follows:

$$\hat{\boldsymbol{x}}_\mathrm{p}[Cn+1 : C(n+1)],\ \hat{\boldsymbol{x}}_\mathrm{d}[Cn+1 : C(n+1)]. \tag{12}$$

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

$$\hat{\boldsymbol{u}}_\mathrm{p}[C(n+1) : C(n+2)-1], \tag{13}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{u}_\mathrm{p}}[C(n+1) : C(n+2)-1]. \tag{14}$$

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 [26]. By inputting the acceleration to the robot, we obtain the next C samples of (12) as follows:

$$\hat{\boldsymbol{x}}_\mathrm{p}[C(n+1)+1 : C(n+2)], \tag{15}$$

$$\hat{\boldsymbol{x}}_\mathrm{d}[C(n+1)+1 : C(n+2)], \tag{16}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{p}}[C(n+1)+1 : C(n+2)], \tag{17}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{d}}[C(n+1)+1 : C(n+2)]. \tag{18}$$

Here, $\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{p}}$ and $\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{d}}$ are calculated based on the state equations (3) and (4) as follows:

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{p}}[k+1] = \boldsymbol{A}\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{p}}[k]\boldsymbol{A}^\top + \boldsymbol{B}\hat{\boldsymbol{S}}_{\boldsymbol{u}_\mathrm{p}}[k]\boldsymbol{B}^\top, \tag{19}$$

$$\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{d}}[k+1] = \boldsymbol{A}\hat{\boldsymbol{S}}_{\boldsymbol{x}_\mathrm{d}}[k]\boldsymbol{A}^\top + \boldsymbol{B}\hat{\boldsymbol{S}}_{\boldsymbol{u}_\mathrm{d}}[k]\boldsymbol{B}^\top. \tag{20}$$

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

III. Training Method

A. 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:

$$L = \frac{1}{K-1}\sum_{k=1}^{K-1}(\alpha l_1[k] + \beta l_2[k] + \gamma l_3[k]), \tag{21}$$

where

$$l_1[k] = \log\left(2\pi\sqrt{\det \hat{\boldsymbol{S}}_{\boldsymbol{p}_\mathrm{d}}[k]}\right) + (\boldsymbol{p}_\mathrm{d}[k] - \hat{\boldsymbol{p}}_\mathrm{d}[k])^\top \hat{\boldsymbol{S}}_{\boldsymbol{p}_\mathrm{d}}^{-1}[k](\boldsymbol{p}_\mathrm{d}[k] - \hat{\boldsymbol{p}}_\mathrm{d}[k]), \tag{22}$$

$$l_2[k] = \max(0, \|\hat{\boldsymbol{p}}_\mathrm{p}[k]\| - p_\mathrm{lim}), \tag{23}$$

$$l_3[k] = \|\hat{\boldsymbol{u}}_\mathrm{p}[k] - \hat{\boldsymbol{u}}_\mathrm{p}[k-1]\|. \tag{24}$$

Here, α, β, and γ 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, we do not have to prepare the trajectories of the frying pan $\boldsymbol{x}_\mathrm{p}$, which realizes the command trajectories $\boldsymbol{x}_\mathrm{d}$. Hence, we can use arbitrary command trajectories such as spline curves generated automatically regardless of the contact model.

B. Training Strategy Based on Curriculum Learning

During the training of the seq2seq models with (21) by backpropagation, the gradient of the training loss L back-propagates through the contact model in (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. We herein applied curriculum learning [22]. In the beginning of the training, we used a simple task: reproducing the positions of the frying pan. We generated

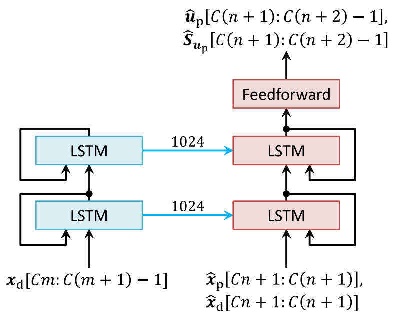

$\hat{\boldsymbol{u}}_\mathrm{p}[C(n+1): C(n+2)-1]$, $\hat{\boldsymbol{S}}_{\boldsymbol{u}_\mathrm{p}}[C(n+1): C(n+2)-1]$
Feedforward
LSTM
1024
LSTM
LSTM
1024
LSTM
$\boldsymbol{x}_\mathrm{d}[Cm: C(m+1)-1]$
$\hat{\boldsymbol{x}}_\mathrm{p}[Cn+1: C(n+1)]$, $\hat{\boldsymbol{x}}_\mathrm{d}[Cn+1: C(n+1)]$

Fig. 5. Implementation of the proposed seq2seq model.

trajectories, $\boldsymbol{x}_\mathrm{p}[0:K-1]$ and $\boldsymbol{x}_\mathrm{d}[0:K-1]$, with a contact model as described in Section II-B. We used the following loss function, L':

$$L' = \frac{1}{K-1}\sum_{k=1}^{K-1}\left[\log\left(2\pi\sqrt{\det \hat{\boldsymbol{S}}_{\boldsymbol{p}_\mathrm{p}}[k]}\right) + (\boldsymbol{p}_\mathrm{p}[k] - \hat{\boldsymbol{p}}_\mathrm{p}[k])^\top \hat{\boldsymbol{S}}_{\boldsymbol{p}_\mathrm{p}}^{-1}[k](\boldsymbol{p}_\mathrm{p}[k] - \hat{\boldsymbol{p}}_\mathrm{p}[k])\right]. \tag{25}$$

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 (22) 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, we can use the contact model for training. By switching to such end-to-end training, we are not required to specify the motion of the frying pan. Therefore, we can use the additional penalties in the loss functions as $l_2[k]$ and $l_3[k]$ in (21).

IV. Implementation

A. Implementation of Seq2seq Model

Here we describe an implementation of the seq2seq model. Fig. 5 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 with 500 units. Each unit in the recurrent layers is implemented as Long Short-Term Memory (LSTM) [27] with tanh activation functions. The seq2seq model receives/generates $C = 100$ samples at a time to reduce the number of RNN iterations [16]. This method can reduce the computational costs for backpropagation through time.

B. Implementation of Training

The training data are generated by simulation. We used two types of command trajectories: 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 II-B. By generating the shaking motion of the frying pan $\boldsymbol{u}_\mathrm{p}[0:K-2]$ and inputting them to the contact model, we

TABLE I
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 $(e-10)/10$ and L with probability $1-(e-10)/10$.	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 $(e-30)/10$ and by B-spline curves with probability $1-(e-30)/10$.
5	41–50	L	Generated by B-spline curves.

obtain the trajectories of the frying pan $\boldsymbol{x}_{\mathrm{p}}[0:K-1]$ and the object $\boldsymbol{x}_{\mathrm{d}}[0:K-1]$. $\boldsymbol{x}_{\mathrm{p}}[0:K-1]$ 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, we obtain the trajectories of the object $\boldsymbol{x}_{\mathrm{d}}[0:K-1]$. 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, we set the friction parameters 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 I. In the loss function L in (21), we set $\alpha = 1$, $\beta = 10$, $\gamma = 1$, and $p_{\mathrm{lim}} = 0.2$ m, respectively. The ratio among α, β, and γ was determined by the approximate amounts of gradients of each term as follows:

$$\frac{\partial l_1[k]}{\partial \boldsymbol{u}_{\mathrm{p}}[k-1]} \approx 2(\boldsymbol{p}_{\mathrm{d}}[k] - \hat{\boldsymbol{p}}_{\mathrm{d}}[k])^\top \hat{\boldsymbol{S}}_{p_{\mathrm{d}}}^{-1}[k]\Delta t^2$$
$$\approx 10^{-3} \cdot (10^{-4})^{-1} \cdot (10^{-3})^2 = 10^{-5}, \quad (26)$$

$$\frac{\partial l_2[k]}{\partial \boldsymbol{u}_{\mathrm{p}}[k-1]} \approx \Delta t^2 \approx 10^{-6}, \quad (27)$$

$$\frac{\partial l_3[k]}{\partial \boldsymbol{u}_{\mathrm{p}}[k-1]} \approx 1, \quad (28)$$

where we 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, $\alpha = 1$, $\beta = 10$, and $\gamma = 10^{-5}$ are preferred. However, we set γ to a large value as described above to make the jerk small such that the manipulators can follow the deformed trajectories. We used Adam [28] with default settings to optimize the seq2seq models.

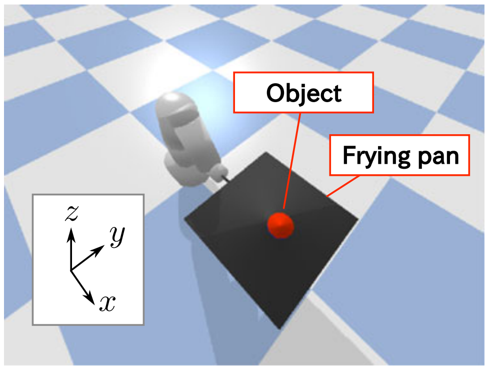


Fig. 6. Overview of the pybullet simulation.

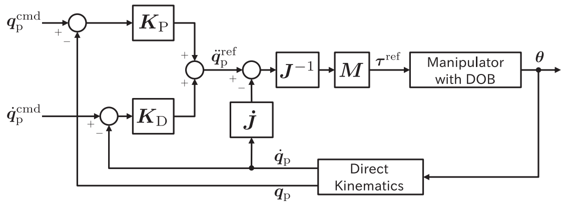


Fig. 7. Control system for the manipulator in pybullet.

V. SIMULATION

A. Implementation of Simulation

We used two simulation environments: the two-dimensional physics model as described in Section II-B and a manipulator in pybullet.

Fig. 6 shows the setup of the simulation in pybullet. We implemented a six degrees of freedom manipulator. 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. 7. This controller controls the position and attitude of the frying pan: x, y, z, roll, pitch, and yaw. A disturbance observer (DOB) is applied to the manipulator. $\boldsymbol{\theta}$ are the joint angles and $\boldsymbol{\tau}^{\mathrm{ref}}$ are the joint torque references. The commands to the PD controller at the k-th sample, $\boldsymbol{q}_{\mathrm{p}}^{\mathrm{cmd}}[k]$, is defined as follows:

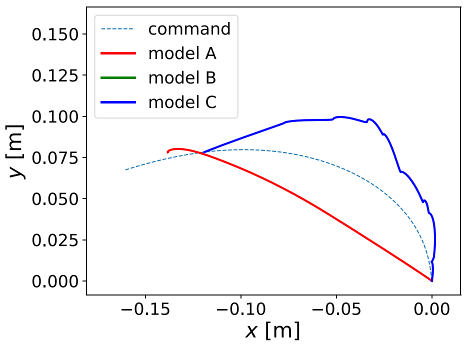
$$\boldsymbol{q}_{\mathrm{p}}^{\mathrm{cmd}}[k] = \left[\hat{x}_{\mathrm{p}}[k] + 0.65, \hat{y}_{\mathrm{p}}[k], 0.3, 0.0, \frac{\pi}{2}, 0.0\right]^\top \quad (29)$$

where $\hat{x}_{\mathrm{p}}[k]$ and $\hat{y}_{\mathrm{p}}[k]$ are calculated by (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.

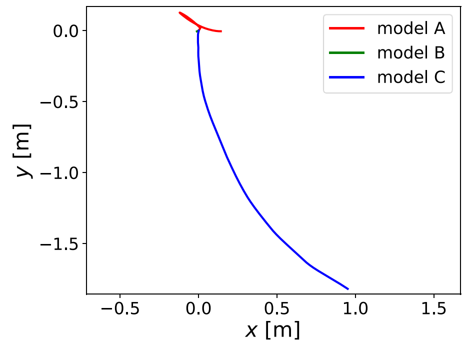
B. Results

We trained three seq2seq models as follows:

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

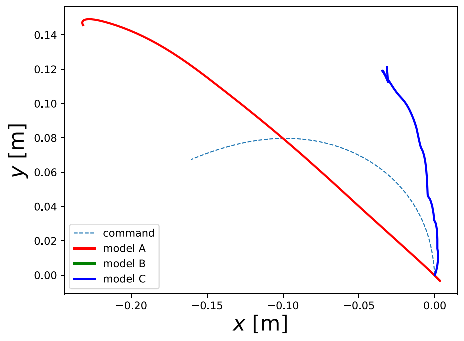


(a) Object positions

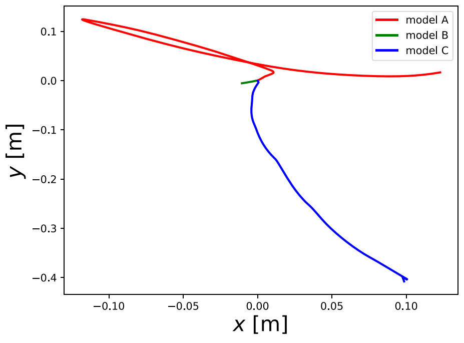


(b) Frying pan positions

Fig. 8. Reproduced trajectories of the object and the frying pan in an ideal situation. (a) Object positions. (b) Frying pan positions.



(a)



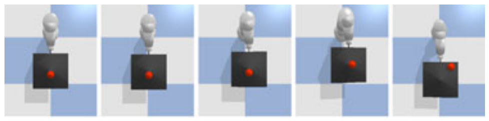
(b)

Fig. 9. Reproduced trajectories of the frying pan and the object in pybullet. In model C (blue curves), the manipulator extended the arm to the limit. (a) Object positions. (b) Frying pan positions.

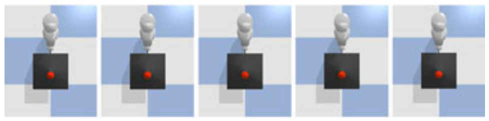
3) A seq2seq model trained with only L': with only stage 1 in Table I for 50 epochs (model C).

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.

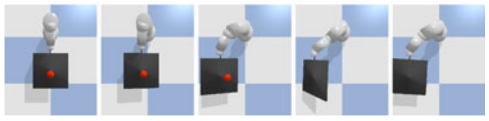
Initially, we evaluated these models in an ideal situation. We used the physics model as explained in Section II-B. It is the same physics model used in the training. Fig. 8 shows the trajectories of the objects and frying pans. All trajectories start from the origin, that is, $\hat{\boldsymbol{p}}_{\mathrm{p}}[0] = \hat{\boldsymbol{p}}_{\mathrm{d}}[0] = [0, 0]^\top$. 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.



(a)



(b)



(c)

Fig. 10. Snapshots for each trained seq2seq model. (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.

Next, we evaluated the performance with the pybullet simulation. 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. 9 shows the reproduced trajectories by the three trained seq2seq models. In addition, Fig. 10 shows the snapshots of the manipulator for each seq2seq model. The length of the

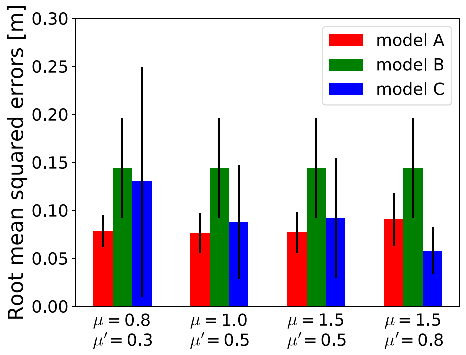


Fig. 11. Reproduction errors of the object by the trained seq2seq models.

command trajectory is set to $K = 1200$. Initially, from the result of model B, we 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 III. 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 (23), 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 (23), model A generated shaking motion of the frying pan within a small range. Thus, from these results, we conclude the proposed curriculum allowed the seq2seq models to learn nonprehensile manipulation tasks including the contact models.

Next, we verified the generalization ability of the trained seq2seq models by the two-dimensional physics model with various friction parameters. Fig. 11 shows the 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 $\hat{\boldsymbol{p}}_\mathrm{d}[0:K-1]$. We calculated 10 trajectories for each parameter. 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, we conclude that the seq2seq model can handle parameter fluctuations that did not exist during training.

VI. Conclusion

This letter 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, we proposed a training curriculum that commences training without the contact models. The proposed curriculum allows the seq2seq models to learn nonprehensile manipulation even if the contact models interfered with the backpropagation. We used the sliding manipulation as an example. We validated the proposed method by a manipulator in a dynamics simulator and two-dimensional physics models. In addition, we 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 apply the proposed method to other types of manipulations. In addition, we would like to investigate the reason that the proposed method makes neural networks robust against changes in physics parameters.

References

[1] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *Int. J. Robot. Res.*, vol. 18, no. 1, pp. 64–92, 1999.

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

[3] P. Lertkultanon and Q. C. Pham, "Dynamic non-prehensile object transportation," in *Proc. 13th Int. Conf. Control Automat. Robot. Vis.*, 2014, pp. 1392–1397.

[4] 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, Mar. 2016.

[5] K. M. Lynch, "Nonprehensile robotic manipulation: Controllability and planning," Ph.D. dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 1996.

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

[7] T. Nierhoff and S. Hirche, "Fast trajectory replanning using Laplacian mesh optimization," in *Proc. 12th Int. Conf. Control Automat. Robot. Vis.*, 2012, pp. 154–159.

[8] 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.

[9] 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.

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

[11] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 5074–5082.

[12] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," Jul. 2017, arXiv:1707.02920.

[13] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real transfer of robotic control with dynamics randomization," Oct. 2017, arXiv:1710.06537.

[14] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural 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] 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. Workshop Sens., Actuation, Motion Control Optim.*, 2017, pp. 1–6.

[17] 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.

[18] 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, Aug. 2009.

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

[20] 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.

[21] 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.

[22] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proc. 26th Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[23] O. Vinyals and Q. V. Le, “A neural conversational model,” in *Proc. 31st Int. Conf. Machine Learn.*, 2015, pp. 1–13.

[24] 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. Natural Lang. Learn.*, 2016, pp. 280–290.

[25] T. Yamada, S. Murata, H. Arie, and T. Ogata, “Representation learning of logic words by an RNN: From word sequences to robot actions,” *Frontier Neurorobot.*, vol. 11, 2017, Art. no. 70.

[26] 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 Netw. Learn. Syst.*, vol. 28, no. 4, pp. 830–848, Apr. 2017.

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

[28] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–13.