Trajectory Adjustment for Nonprehensile Manipulation using Latent Space of Trained Sequence-to-Sequence Model

Kyo Kutsuzawa, Sho Sakaino, and Toshiaki Tsuji

This is an electronic version of an article published in *Advanced Robotics*
Vol. 33, No. 21, pp. 1144–1154, 2019
Advanced Roboticsis available online at: www.tandfonline.com/10.1080/01691864.2019.1673204;
10.1080/01691864.2019.1673204.

FULL PAPER

Trajectory Adjustment for Nonprehensile Manipulation using Latent Space of Trained Sequence-to-Sequence Model

K. Kutsuzawa[a,b*], S. Sakaino[c,d], and T. Tsuji[a]

[a] *Graduate School of Science and Engineering, Saitama University, 255 Shimo-Ohkubo, Sakura-ku, Saitama, Japan*; [b] *JSPS Research Fellow (DC2)*; [c] *Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, Japan*
[d] *JST PRESTO*

(*Received 00 Month 201X; accepted 00 Month 201X*)

When robots are used to manipulate objects in various ways, they often have to consider the dynamic constraint. Machine learning is a good candidate for such complex trajectory planning problems. However, it sometimes does not satisfy the task objectives due to a change in the objective or a lack of guarantee that the objective functions will be satisfied. To overcome this issue, we applied a method of trajectory deformation by using sequence-to-sequence (seq2seq) models. We propose a method of adjusting the generated trajectories, by utilizing the architecture of seq2seq models. The proposed method optimizes the latent variables of the seq2seq models instead of the trajectories to minimize the given objective functions. The verification results show that the use of latent variables can obtain the desired trajectories faster than direct optimization of the trajectories.

Keywords: nonprehensile manipulation; sequence-to-sequence model; latent variable; neural network; motion optimization

1. Introduction

As robots are increasingly used in a wide range of applications, they will often have to consider the dynamic constraint when they manipulate objects. For example, in the task of moving objects on a spatula[1, 2], the robots should apply appropriate acceleration to avoid dropping the objects. In the task of sliding ingredients on a plate[3, 4], the robots have to get the ingredients to move between the static friction and dynamic friction. For such kind of manipulation, called nonprehensile manipulation[5], the robots should satisfy the dynamic constraint while achieving the other operation goals such as reaching the final position, fitting the motions within the range of motion, and limiting the velocity within the limitations.

It is generally difficult to generate motions that satisfy the dynamic constraint as well as accomplish the operational goals. Kinodynamic motion planning[6], 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)[7, 8] and Model Predictive Control (MPC)[2]. Lertkultanon *et al.*[8] realized a robot capable of manipulating objects on a plate without dropping and avoiding obstacles by using Admissible Velocity Propagation-RRT. Woodruff *et al.* [9] realized an object manipulation task that involved rolling, sliding, and throwing. 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

*Corresponding author. Email: k.kutsuzawa.430@ms.saitama-u.ac.jp

tasks such as contact motions.

To avoid the increase in complexity of trajectory optimization, approaches based on machine learning, especially neural networks, have been researched recently. Levine *et al.* [10] realized various assembly tasks including tight-fitting contact by using neural networks. Yuan *et al.* [11] realized a planar pushing task with obstacle avoidance by using reinforcement learning. Mordatch *et al.* [12] 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.* [13] showed that neural networks trained by MPC can reduce the computation cost. Similarly, Furuta *et al.* [14] 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. In such cases, we need 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[15–18], all of them used domain-specific algorithms. On the other hand, we use intermediate representations of trained neural networks, which are also called latent representations or latent variables. 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 variables is demonstrated in the field of computer vision. The use of gradient descent methods to generate images resulted in unrealistic images[19], whereas the use of gradient descent methods to latent representations of image classification models could obtain realistic images[20]. Recently, some studies used latent representations of motions in the field of reinforcement learning[21–23]. 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.

This paper proposes a method of adjusting the trajectories generated by neural networks by using the latent representations of the trajectories. As neural networks for trajectory generation via latent representations, we use sequence-to-sequence (seq2seq) models[24–28]. Seq2seq models can be used to generate trajectories considering the dynamics by deforming the given trajectories[26, 28]. By optimizing their latent variables, 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 variables can be optimized by using the simplest gradient descent method.

The rest of this paper is organized into the following sections. Section 2 introduces seq2seq models and their application to trajectory deformation. Section 3 explains the proposed method. Section 4 describes the simulation that is conducted to evaluate the proposed method. Section 5 describes an experiment based on the proposed method. Finally, Section 6 concludes this paper.

2. Background

2.1 *Sequence-to-sequence model*

Sequence-to-sequence (seq2seq) models[24, 25] are neural networks used for time-series conversion. Seq2seq models are used in a wide variety of applications such as natural language transla-

tion, text summarization[29], and forming associations between language commands and robot motions[30]. A seq2seq model consists of two recurrent neural networks (RNNs) as illustrated in Fig. 1. The input side RNN is called *encoder*, while the output side RNN is called *decoder*. Seq2seq models have almost no limitation for the construction of a time-series.

Here, we explain the conversion process of seq2seq models in machine translation. First, the encoder receives a sentence in the source language word by word. After the encoder receives the final word, the encoder generates a latent variable. Then, the decoder receives the latent variable and the first word, and generates the next word in the target language. Usually, the special token "BOS," which indicates the beginning of the sentence, is used as the first word. The decoder repeats the process of receiving a word and generating the next word. The generation process of the decoder is completed by generating a special token "EOS," which indicates the end of the sentence. The training progresses in a manner of end-to-end learning; the accuracy of the generated sentence is evaluated and backpropagated to the entire seq2seq model.

One feature of seq2seq models is that the time-series conversion progresses via latent representations of the time-series. The latent representations are considered to represent the features of the time-series for the trained tasks. For example, in natural language translation, it is considered that the latent representation represents the meaning of the sentence [25]. This can be utilized for translating between multiple languages [31] and generating a sentence that has an intermidiate meaning between two sentences[32].

2.2 *Seq2seq model for trajectory deformation*

Seq2seq models can also be applied to nonprehensile manipulation[26–28]. These seq2seq models deform the given trajectories to satisfy the dynamic constraint of the task. They can be applied even if the task includes discontinuous contact models, which are difficult to handle analytically[27, 28].

An overview of the seq2seq model is shown in Fig. 2. This model receives an arbitrary trajectory,

$$X_{\text{in}} = (\boldsymbol{x}_{\text{in}}[0], \cdots, \boldsymbol{x}_{\text{in}}[K-1]), \tag{1}$$

where $\boldsymbol{x}_{\text{in}}[k]$ indicates a state variable at the k-th sample, and K is the length of the sequence. The encoder converts X_{in} into a latent variable $\boldsymbol{z}$. Then, the decoder receives $\boldsymbol{z}$ and generates a sequence of control inputs for a robot as follows:

$$U_{\text{out}} = (\boldsymbol{u}_{\text{out}}[0], \cdots, \boldsymbol{u}_{\text{out}}[K-1], \boldsymbol{\sigma}_{u_{\text{out}}}[0], \cdots, \boldsymbol{\sigma}_{u_{\text{out}}}[K-1]). \tag{2}$$

editedpwhereHere, $\boldsymbol{u}_{\text{out}}[k]$ and $\boldsymbol{\sigma}_{u_{\text{out}}}[k]$ indicate a control input for the robot and its covariance matrix at the k-th sample, respectively. The covariance matrices are used only for training [33]. By inputting U_{out} into the physics model, we obtain a deformed trajectory

$$X_{\text{out}} = (\boldsymbol{x}_{\text{out}}[0], \cdots, \boldsymbol{x}_{\text{out}}[K-1], \boldsymbol{\sigma}_{x_{\text{out}}}[0], \cdots, \boldsymbol{\sigma}_{x_{\text{out}}}[K-1]). \tag{3}$$

Here, $\boldsymbol{x}_{\text{out}}[k]$ and $\boldsymbol{\sigma}_{x_{\text{out}}}[k]$ indicate a state variable and its covariance matrix for the output sequence at the k-th sample, respectively.

The training objective is to deform the given trajectories in such a way that they satisfy the dynamic constraint while retaining the original shapes. The training procedure is summarized as follows; the details are explained in [26]. The model is trained so as to minimize the objective function L for the input sequence X_{in} and output sequences X_{out} and U_{out}:

$$L(X_{\text{out}}, U_{\text{out}}, X_{\text{in}}) = L_1(X_{\text{out}}, X_{\text{in}}) + \alpha L_2(X_{\text{out}}, U_{\text{out}}), \tag{4}$$

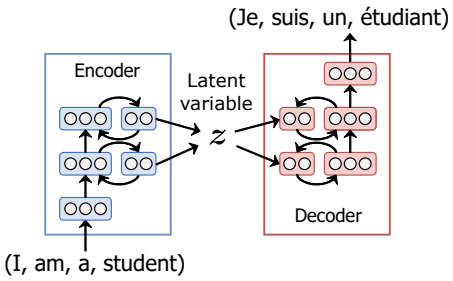


Figure 1. Sequence-to-sequence model. This figure shows an example of machine translation from an English sentence into a French sentence. The encoder receives the source sentence word by word and generates a latent variable $\boldsymbol{z}$. The decoder receives the latent variable $\boldsymbol{z}$ and generates the target sentence word by word.

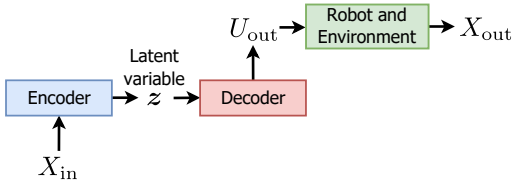


Figure 2. Seq2seq model for trajectory deformation. The encoder receives the original trajectory and generates a latent variable $\boldsymbol{z}$. The decoder receives the latent variable $\boldsymbol{z}$ and generates control inputs. By inputting the control inputs into the robot, a deformed trajectory is obtained.

Here, L_1 indicates the reproduction loss between X_{in} and X_{out}, and is expressed as follows:

$$L_1(X_{\text{out}}, X_{\text{in}}) = -\sum_{k=0}^{K-1} \log p(\boldsymbol{D}\boldsymbol{x}_{\text{in}}[k] | \boldsymbol{D}\boldsymbol{x}_{\text{out}}[k], \boldsymbol{D}\sigma_{\boldsymbol{x}_{\text{out}}}[k]\boldsymbol{D}^\top). \tag{5}$$

Here, $\boldsymbol{D}$ is a constant diagonal matrix that is used to align the scales between different units such as meters and radians, and $p(x|\mu, \sigma)$ indicates the probability of x under a Gaussian distribution with mean μ and covariance σ. L_2 is the penalty for the case in which the output motion exceeds the dynamic constraint, and α is the weight coefficient. If α is large enough, L_2 converges to zero while L_1 decreases. Finally, the model outputs a trajectory that satisfies the dynamic constraint while retaining the original shape of the input trajecotry.

3. Method

3.1 *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, we often have 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, we should decrease the velocity 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, we propose a method of adjusting the deformed trajectories by utilizing the architecture of seq2seq models.

3.2 *Optimization of the latent variable*

The most straight-forward approach is to optimize the trajectories directly with the given objective functions. There have been many studies on trajectory adjustment[15–18]; however, this

approach may be difficult because of the large number of dimensions.

Instead of utilizing this approach, we aim to use latent variables for optimization. As explained in Section 2.1, the latent variables represent the features of the trained time-series in a low-dimensional space. In this task, the latent variables are expected to represent the trajectories that satisfy the dynamic constraint. Therefore, we expect that the use of latent variables are beneficial to such adjustment tasks. In this paper, we aim to adjust the latent variables instead of the trajectories.

To optimize the latent variables, we need to quantify the relationship between the latent variables and the deformed trajectories. This relationship is non-obvious; however, we can associate their small displacements 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, we can obtain the gradient of the objective function J with respect to the latent variables $\boldsymbol{z}$ as follows:

$$\frac{\partial}{\partial \boldsymbol{z}} J(X_{\text{out}}, U_{\text{out}}) = \frac{\partial U_{\text{out}}}{\partial \boldsymbol{z}} \frac{\partial}{\partial U_{\text{out}}} J(X_{\text{out}}, U_{\text{out}}). \tag{6}$$

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

$$\boldsymbol{z} \leftarrow \boldsymbol{z} - \eta \frac{\partial}{\partial \boldsymbol{z}} J(X_{\text{out}}, U_{\text{out}}), \tag{7}$$

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 variables. 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 (7).

3.3 *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 (6).
step 6 Update $\boldsymbol{z}$ by using (7).
step 7 Repeat 3–6 until convergence.

Finally, we obtain an optimized $\boldsymbol{z}$ and its corresponding trajectory X_{out}. An overview of this procedure is illustrated in Fig. 3.

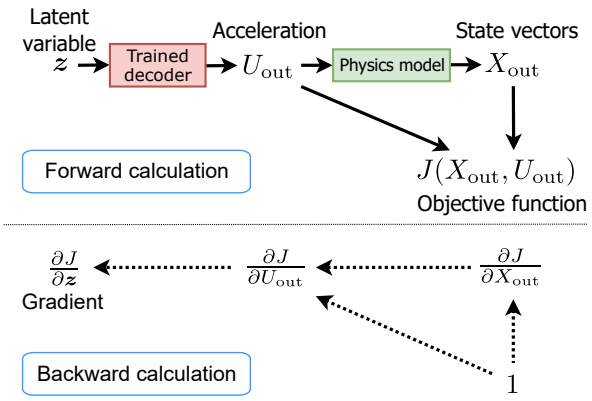


Figure 3. Overview of the proposed method. Top: the forward calculation graph. Bottom: the backward calculation graph.

4. Simulation

4.1 *Physics model*

In this paper, we consider the task of turning over pancakes with a spatula, which is based on planar physics. The overview of the physics model is illustrated in Fig. 4. A pancake is placed on the spatula, and the pancake is manipulated without slipping or dropping by moving the spatula.

Let the state variable at the k-th sample be $\boldsymbol{x}[k]$, and let the acceleration input to the spatula at the k-th sample be $\boldsymbol{u}[k]$. Then, the state equation of the spatula is expressed as follows:

$$\boldsymbol{x}[k+1] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k]. \tag{8}$$

The state variable $\boldsymbol{x}[k]$, the acceleration input $\boldsymbol{u}[k]$, and matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are defined as follows:

$$\boldsymbol{x}[k] = [y[k], z[k], \theta[k], v_y[k], v_z[k], v_\theta[k]]^\top, \tag{9}$$

$$\boldsymbol{u}[k] = [a_y[k], a_z[k], a_\theta[k]]^\top, \tag{10}$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_3 & \Delta t \boldsymbol{I}_3 \\ \boldsymbol{O}_3 & \boldsymbol{I}_3 \end{bmatrix}, \tag{11}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{\Delta t^2}{2} \boldsymbol{I}_3 \\ \Delta t \boldsymbol{I}_3 \end{bmatrix}. \tag{12}$$

Here, $y[k]$ and $z[k]$ indicate the position of the spatula and $\theta[k]$ is the attitude of the spatula. $v_\bullet[k]$ and $a_\bullet[k]$ are the velocity and the acceleration, respectively. $\boldsymbol{I}_3 \in \mathrm{R}^{3\times3}$ and $\boldsymbol{O}_3 \in \mathrm{R}^{3\times3}$ denote the identity matrix and zero matrix, respectively. Δt is the sampling interval; we set $\Delta t = 1$ ms. In addition, we represent the position and velocity of the spatula, $\boldsymbol{p}[k]$ and $\dot{\boldsymbol{p}}[k]$, respectively, as follows:

$$\boldsymbol{p}[k] = [y[k], z[k], \theta[k]]^\top, \tag{13}$$

$$\dot{\boldsymbol{p}}[k] = [v_y[k], v_z[k], v_\theta[k]]^\top. \tag{14}$$

The pancake is fixed to the spatula while the dynamic constraint is maintained. The condition of the dynamic constraint is expressed as follows:

$$|\phi[k]| < \arctan \mu_{\max}. \tag{15}$$

Here, $\mu_{\max}$ is the static friction coefficient; we let $\mu_{\max} = 1.0$. $\phi[k]$ denote the argument angle of

the resultant force of the inertial force and the gravity with respect to the spatula as follows:

$$\phi[k] \equiv \arg\left(\boldsymbol{g} - \boldsymbol{u}[k]\right) - \theta[k]. \tag{16}$$

Here, arg indicates the angle of vectors on the y-z plane, $\boldsymbol{g} = [0, -9.8, 0]^\top$ is the gravity, and $\theta[k]$ is the inclination angle of the spatula.

4.2 *Implementation of seq2seq model*

The training procedure and the model architecture followed those in the previous work [26]; however we used two layers of LSTMs with 64 units as the encoder and the decoder. Thus, the latent variable $\boldsymbol{z}$ has $64 \times 2 = 128$ dimensions. It is more than 10 times smaller than that of the trajectories; as explained in Section 4.3.1, each trajectory has $3 \times 500 = 1500$ dimensions.

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

4.3 *Adjusting trajectories to reach the given end positions*

We generated trajectories for reaching the given end positions by using the trained seq2seq model.

4.3.1 Objective function

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

$$J_1(X_{\text{out}}, U_{\text{out}}) \equiv \|\boldsymbol{D}(\boldsymbol{p}[K-1] - \boldsymbol{p}_{\text{end}})\|^2 + \alpha f_{\text{dyn}}(X_{\text{out}}, U_{\text{out}}), \tag{17}$$

where, $\boldsymbol{p}_{\text{end}}$ denotes the desired end position, and α is a constant value; here, we set $\alpha = 1$. $\boldsymbol{D} = \text{diag}[1, 1, 0.1]$ 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:

$$f_{\text{dyn}}(X_{\text{out}}, U_{\text{out}}) \equiv \sum_{k=0}^{K-1} \max(|\phi[k]| - \arctan \mu_{\max}, 0). \tag{18}$$

Here, max indicates the maximum function. This penalty increases when (15) 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 L_2 in (4). Even if the seq2seq models are trained correctly, we need such a penalty to satisfy the dynamic constraint. The reason is that not all points in the latent space are associated with valid trajectories. Latent variables 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 variables have a dense and continuous distribution in a certain region, we can find invalid latent variables that were not generated by the encoder during optimization.

We used various patterns of end positions $\boldsymbol{p}_{\text{end}} = [y_{\text{end}}, z_{\text{end}}, \theta_{\text{end}}]$ from the following range:

$$-0.1 \text{ m} \leq y_{\text{end}} \leq 0.1 \text{ m}, \tag{19}$$

$$-0.1 \text{ m} \leq z_{\text{end}} \leq 0.1 \text{ m}, \tag{20}$$

$$-\pi \text{ rad} \leq \theta_{\text{end}} \leq \pi \text{ rad}. \tag{21}$$

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

4.3.2 Original trajectories

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

$$\hat{J}_1(X_{\text{out}}, U_{\text{out}}) \equiv \|\boldsymbol{D}(\boldsymbol{p}[K-1] - \boldsymbol{p}_{\text{end}})\|^2 \tag{22}$$

This is the first term of (17). Since this objective function considers only the kinematics, it is easy to design. We used S-curve acceleration/deceleration trajectories from the initial position to the end position.

4.3.3 Results

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

case 1 Updating the latent variable $\boldsymbol{z}$ starting with the encoder result (i.e., the proposed method).
case 2 Updating the latent variable $\boldsymbol{z}$ 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 variables 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 variable $\boldsymbol{z}_0$ was sampled from the Gaussian distribution with mean $\boldsymbol{0}$ and variance $\boldsymbol{I}$.

In general, the distribution of the latent variables is not a standard Gaussian distribution. The actual distribution may be complex and is hard to identify. Therefore, in **case 2**, we used the standard Gaussian distribution. Even though we can extend seq2seq models to specify the distribution of the latent variables [32], such models often fail to learn various trajectories.

The progress of the objective function values is shown in Fig. 5. Here, we set the learning coefficient to $\eta = 200, 200, \text{and } 1000$ 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 variables 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. 6, 7, and 8. 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. 9. The RMSEs in **case 1** were 0.27 mm and 0.30 deg on average. These results were comparable to the neural networks in [14], 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.

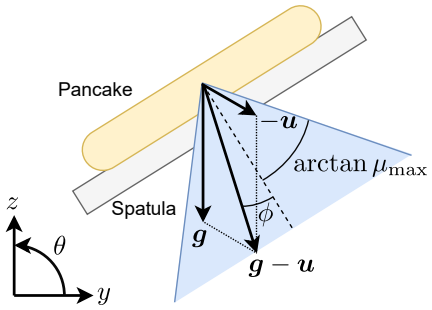


Figure 4. Physics model.

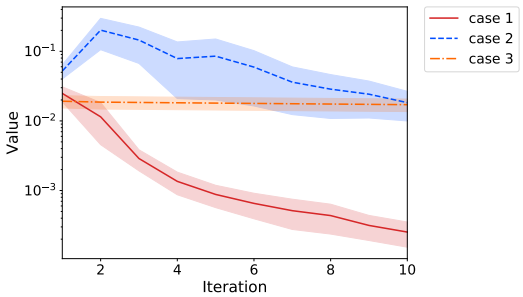


Figure 5. Progress of the objective functions. The lines indicate the mean losses of 27 trials with various end positions. The filled areas indicate their 95 % confident levels.

5. Experiment

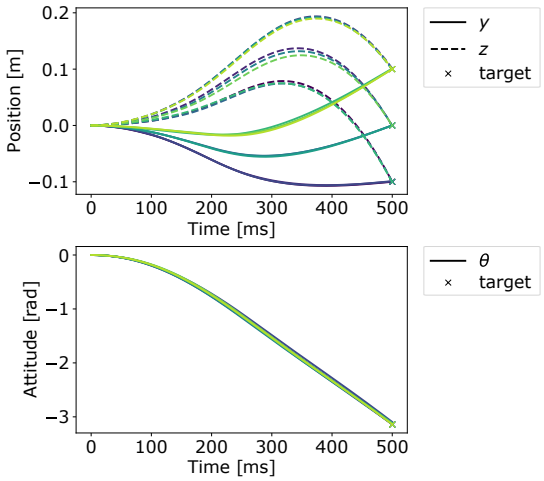
5.1 *Setup of the robot*

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

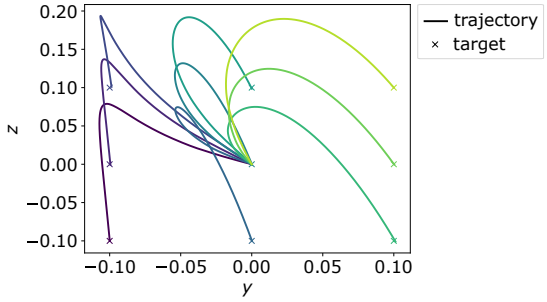
A P-D controller with disturbance observer (DOB) [34] is implemented to control the tip of the spatula position and attitude. The control system is illustrated in Fig. 10. Here, $\boldsymbol{q}$ indicates the six-dimensional position/attitude of the spatula in the Cartesian coordinate system. The control 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:

$$\boldsymbol{q}^{\mathrm{cmd}} = \left[0.5, y[k], z[k] + 0.35, \theta[k], \frac{\pi}{2}, 0\right]^\top. \tag{23}$$

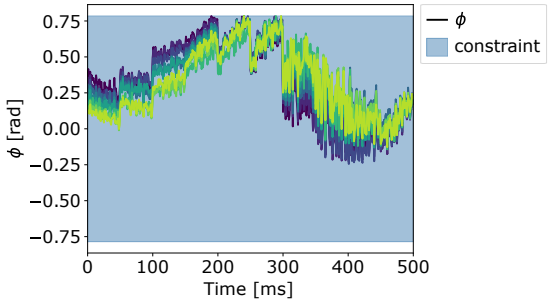
Here, we did not use the generated acceleration $\boldsymbol{u}[k]$ directly to avoid drift errors, which cause large errors in the final positions. Even though the control commands were the position and attitude, the DOB forced the robot to follow the acceleration reference, $\ddot{\boldsymbol{q}}^{\mathrm{ref}}$, 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. $\boldsymbol{K}_{\mathrm{p}} = \mathrm{diag}[K_{\mathrm{pp}}, K_{\mathrm{pp}}, K_{\mathrm{pp}}, K_{\mathrm{pr}}, K_{\mathrm{pr}}, K_{\mathrm{pr}}]$, $\boldsymbol{K}_{\mathrm{v}} = \mathrm{diag}[K_{\mathrm{vp}}, K_{\mathrm{vp}}, K_{\mathrm{vp}}, K_{\mathrm{vr}}, K_{\mathrm{vr}}, K_{\mathrm{vr}}]$, $\boldsymbol{J}$, 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_{vp}, and derivative attitude gain K_{vr} 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.



(a) Time-position



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

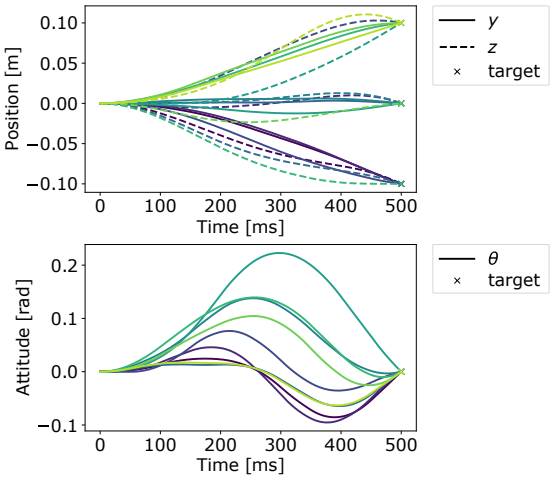


(c) Angle of the contact force of the pancake with respect to the spatula. The dynamic constraint is satisfied in the blue area.

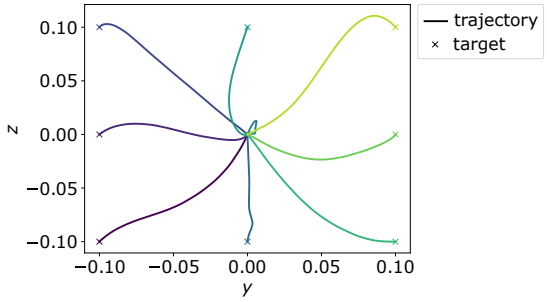
Figure 6. Trajectories obtained after optimization by the proposed method when the final attitude is $-\pi$.

5.2 *Trajectory*

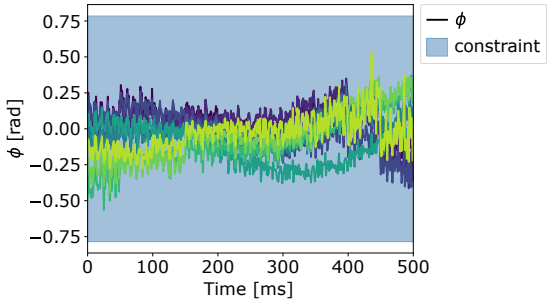
While executing the turning over motions, the manipulator may fail in the task if the velocity is too large. Therefore, we should adjust the trajectories to reduce the velocity. To obtain such



(a) Time-position



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

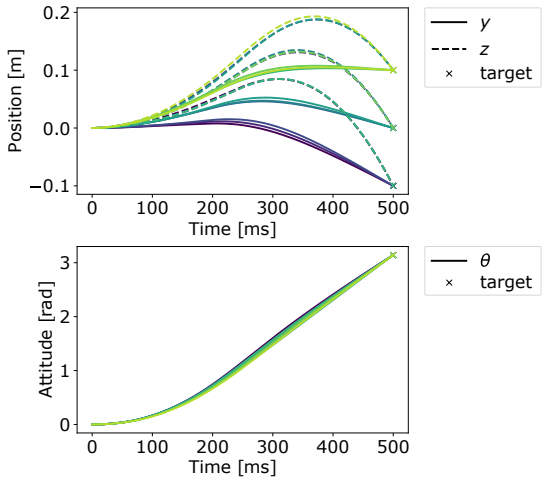


(c) Angle of the contact force of the pancake with respect to the spatula. The dynamic constraint is satisfied in the blue area.

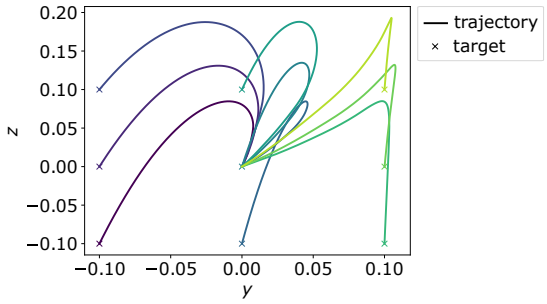
Figure 7. Trajectories obtained after optimization by the proposed method when the final attitude is 0.

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

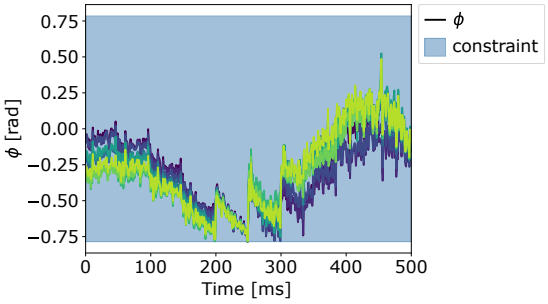
$$
\begin{aligned}
J_2(X_{\text{out}}, U_{\text{out}}) \equiv & \max\left(\sqrt{y^2[K-1] + z^2[K-1]} - p_{\text{lim}}, 0\right) \\
& + 0.1 \max\left(|\theta[K-1] - \theta_{\text{end}}| - \theta_{\text{lim}}, 0\right) \\
& + \alpha f_{\text{dyn}}(X_{\text{out}}, U_{\text{out}}) + \frac{\beta}{K} \sum_{k=0}^{K-1} \| \max(0, \boldsymbol{D}(\text{abs}(\dot{\boldsymbol{p}}[k]) - \boldsymbol{v}_{\text{lim}}))\|^2. \qquad (24)
\end{aligned}
$$



(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 8. Trajectories obtained after optimization by the proposed method when the final attitude is π.

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; we set these as $-\frac{3}{4}\pi$ rad, 0.2 m, and 0.5 rad, respectively. α and β are constant values; here we set $\alpha = 1$ and $\beta = 0.01$. $\boldsymbol{v}_{\text{lim}}$ indicates the desired range of the velocity; we set $\boldsymbol{v}_{\text{lim}} = [1.5, 1.5, 4.5]^\top$. 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

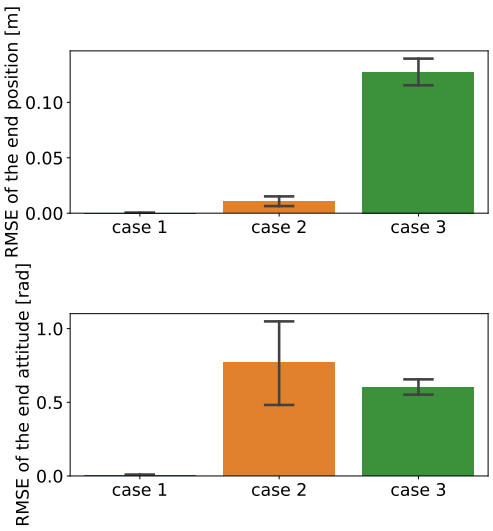


Figure 9. Root mean square errors (RMSEs) of the end positions. The error bars indicate their 95 % confident levels.

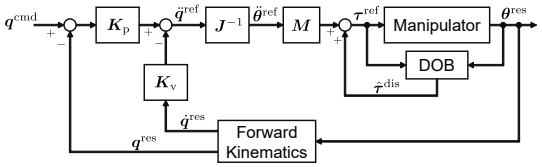


Figure 10. Control system.

$\eta = 100$.

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

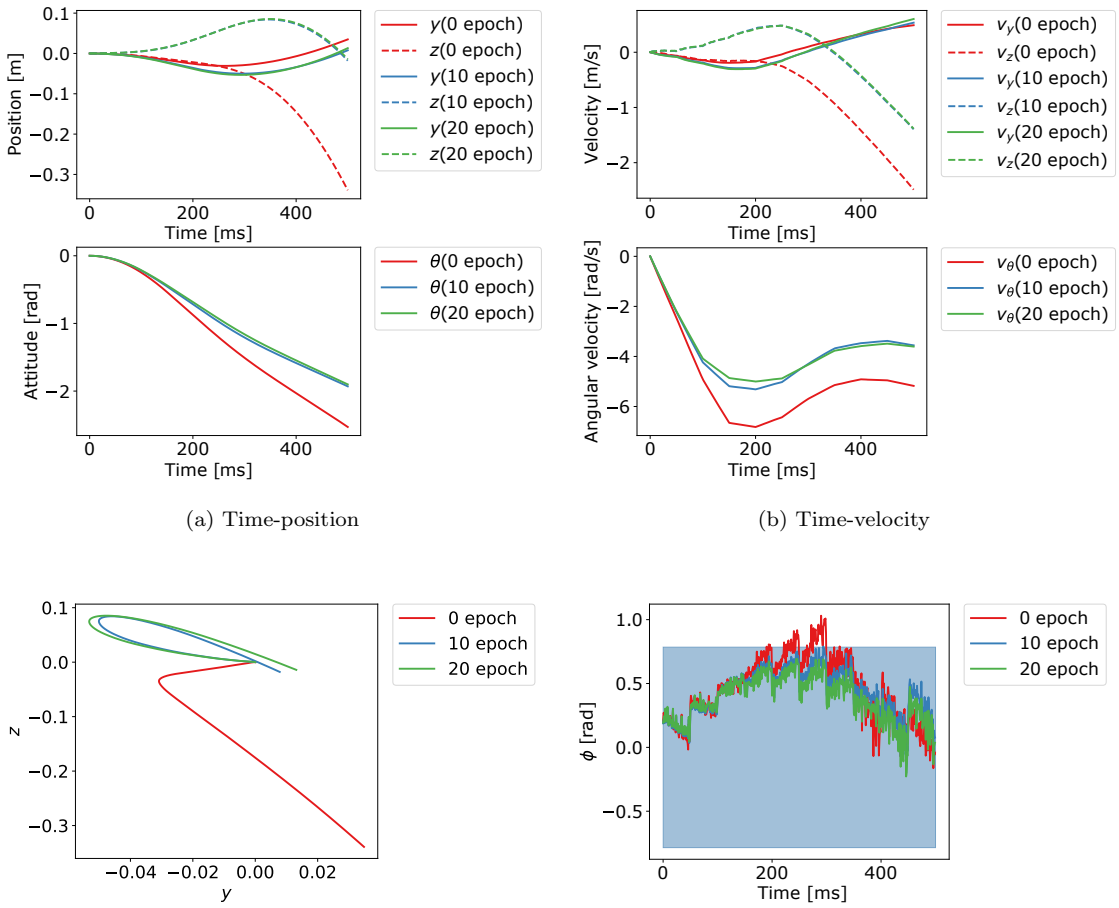
5.3 *Result*

The optimization results in the 0th, 10th and 20th iterations are shown in Fig. 11. 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 variable 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. 12. 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. 13.

6. Conclusion

Trajectory generation using neural networks can deal with complex tasks such as kinodynamic motion planning, but sometimes they fail to achieve the objectives accurately. Therefore, adjustment of the generated trajectories is necessary. In this paper, we proposed a method of adjusting the trajectories generated by seq2seq models for trajectory deformation. Seq2seq models can convert trajectories to satisfy dynamic constraints via latent variables representing the features of the trajectories. The proposed method optimizes the latent variables instead of trajectories to



(a) Time-position

(b) Time-velocity

(c) Trajectories in the y-z plane.

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

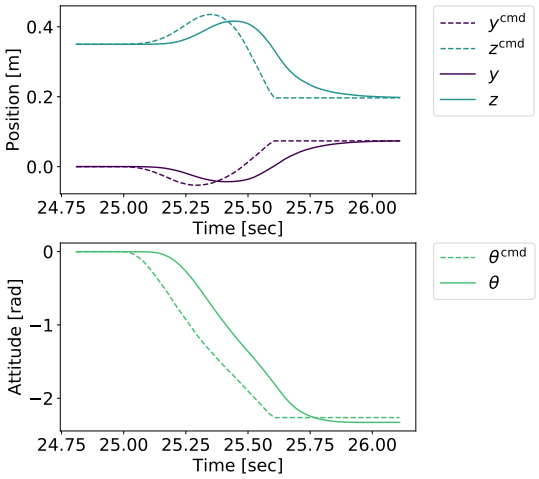
Figure 11. Trajectories obtained after optimization with J_2.

minimize the given objective functions. Through simulation, we verified that the use of latent variables 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.

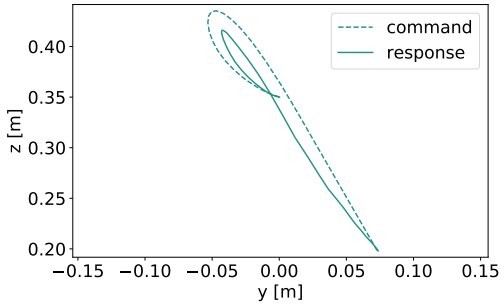
We expect that the proposed approach, which considers the latent variables of trajectory generation models that can deal with task-specific constraints, is effective for complex tasks that consist of various constraints and objectives.

Acknowledgement

This work was supported by Grant-in-Aid for JSPS Research Fellow, No. 18J14272.



(a) Position and attitude.



(b) Trajectories in the y-z plane.

Figure 12. Trajectories obtained after optimization by the proposed method with $\boldsymbol{p}_{\mathrm{end1}}$.

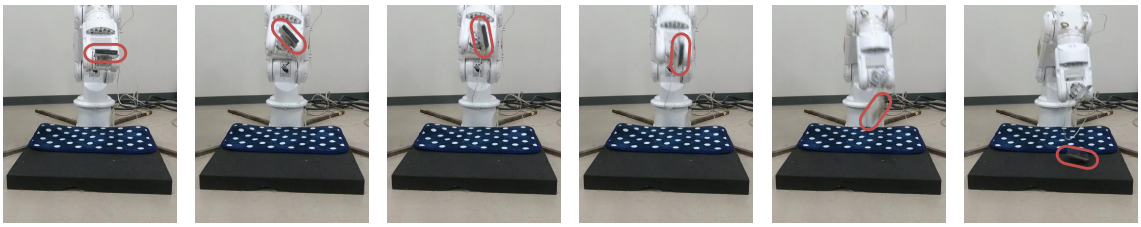


Figure 13. Snapshots of the turning over motion.

References

[1] Tsuji T, Ohkuma J, Sakaino S. Dynamic object manipulation considering contact condition of robot with tool. IEEE Trans Ind Electron. 2016;63(3):1972–1980.

[2] Tsuji T, Kutsuzawa K, Sakaino S. Optimized Trajectory Generation based on Model Predictive Control for Turning Over Pancakes. IEEJ J Ind Appl. 2018;7(1):22–28.

[3] Higashimori M, Utsumi K, Omoto Y, Kaneko M. Dynamic Manipulation Inspired by the Handling of a Pizza Peel. IEEE Trans Robot. 2009;25(4):829–838.

[4] Vose TH, Umbanhowar P, Lynch KM. Sliding manipulation of rigid bodies on a controlled 6-DoF plate. Int J Rob Res. 2012;31(7):819–838.

[5] Lynch KM. Nonprehensile Robotic Manipulation : Controllability and Planning. [Ph.D. thesis]. Carnegie Mellon University. 1996.

[6] Donald B, Xavier P, Canny J, Reif J. Kinodynamic Motion Planning. J ACM. 1993;40(5):1048–1066.

[7] LaValle SM, Kuffner JJ. Randomized kinodynamic planning. Int J Rob Res. 2001;20(5):378–400.

[8] Lertkultanon P, Pham QC. Dynamic non-prehensile object transportation. In: Proc. int. conf. control autom. robot. vis.. 2014. p. 1392–1397.

[9] Woodruff JZ, Lynch KM. Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks. In: Proc. ieee int. conf. robot. autom.. 2017. p. 4066–4073.

[10] Levine S, Wagener N, Abbeel P. Learning Contact-Rich Manipulation Skills with Guided Policy Search. In: Proc. ieee int. conf. robot. autom.. 2015. p. 156–163.

[11] Yuan W, Stork JA, Kragic D, Wang MY, Hang K. Rearrangement with Nonprehensile Manipulation Using Deep Reinforcement Learning. In: 2018 ieee int. conf. robot. autom.. 2018 mar. p. 270–277.

[12] Mordatch I, Lowrey K, Andrew G, Popovic Z, Todorov E. Interactive Control of Diverse Complex Characters with Neural Networks. In: Adv. neural inf. process. syst.. 2015. p. 1–8.

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

[14] Furuta D, Kutsuzawa K, Okamoto T, Sakaino S, Tsuji T. Model Predictive Control based Deep Neural Network for Dynamic Manipulation. In: Proc. annu. conf. ieee ind. electron. soc.. 2017. p. 5215–5220.

[15] Kurniawati H, Fraichard T. From path to trajectory deformation. In: Proc. ieee/rsj int. conf. intell. robot. syst.. 2007. p. 159–164.

[16] Lamiraux F, Bonnafous D. Reactive trajectory deformation for nonholonomic systems: application to mobile robots. In: Proc. ieee int. conf. robot. autom.. Vol. 3. 2002. p. 3099–3104.

[17] Nierhoff T, Hirche S. Fast trajectory replanning using Laplacian mesh optimization. In: Proc. int. conf. control. autom. robot. vis.. 2012. p. 154–159.

[18] Pekarovskiy A, Nierhoff T, Schenek J, Nakamura Y, Hirche S, Buss M. Online deformation of optimal trajectories for constrained nonprehensile manipulation. In: Proc. ieee/rsj int. conf. intell. robot. syst.. 2015. p. 2481–2487.

[19] Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In: Proc. ieee comput. vis. pattern recognit.. 2015 dec. p. 427–436.

[20] Nguyen A, Dosovitskiy A, Yosinski J, Brox T, Clune J. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Adv. neural inf. process. syst. 29. 2016. p. 1–9.

[21] Co-Reyes JD, Liu Y, Gupta A, Eysenbach B, Abbeel P, Levine S. Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings. In: Proc. 35th int. conf. mach. learn.. Vol. 80. 2018 jun. p. 1009–1018. 1806.02813. Available from: http://arxiv.org/abs/1806.02813.

[22] Pahic R, Loncarevic Z, Ude A, Nemec B, Gams A. User Feedback in Latent Space Robotic Skill Learning. IEEE-RAS Int Conf Humanoid Robot. 2018;2018-Novem:791–797.

[23] Lynch C, Khansari M, Xiao T, Kumar V, Tompson J, Levine S, Sermanet P. Learning Latent Plans from Play. arXiv Prepr arXiv190301973. 2019 mar;1903.01973. Available from: https://arxiv.org/abs/1903.01973.

[24] Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proc. conf. empir. methods nat. lang. process.. 2014. p. 1724–1734.

[25] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Proc. int. conf. neural inf. process. syst.. 2014. p. 3104–3112.

[26] Kutsuzawa K, Sakaino S, Tsuji T. Sequence-to-Sequence Models for Trajectory Deformation of Dy-

namic Manipulation. In: Proc. annu. conf. ieee ind. electron. soc.. 2017. p. 5227–5232.
[27] Kutsuzawa K, Sakaino S, Tsuji T. Deformation of Contact Motion by Neural Networks to Adapt for Various Environment Change. In: Proc. ieej int. work. sensing, actuation, motion control. optim.. 2018.
[28] Kutsuzawa K, Sakaino S, Tsuji T. Sequence-to-Sequence Model for Trajectory Planning of Nonprehensile Manipulation Including Contact Model. IEEE Robot Autom Lett. 2018;3(4):3606–3613.
[29] Nallapati R, Zhou B, dos Santos CN, Gülçehre Ç, Xiang B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: Proc. signll conf. comput. nat. lang. learn.. 2016. p. 280–290.
[30] Yamada T, Murata S, Arie H, Ogata T. Dynamical integration of language and behavior in a recurrent neural network for Human-Robot interaction. Front Neurorobot. 2016;10(5):1–17.
[31] Johnson M, Schuster M, Le QV, Krikun M, Wu Y, Chen Z, Thorat N, Viégas F, Wattenberg M, Corrado G, Hughes M, Dean J. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. arXiv Prepr arXiv161104558. 2016 nov;.
[32] Bowman SR, Vilnis L, Vinyals O, Dai AM, Jozefowicz R, Bengio S. Generating Sentences from a Continuous Space. arXiv Prepr arXiv151106349. 2015 nov;.
[33] Murata S, Yamashita Y, Arie H, Ogata T, Sugano S, Tani J. Learning to perceive the world as probabilistic or deterministic via interaction with others: a neuro-robotics experiment. IEEE Trans Neural Networks Learn Syst. 2015;28(4):830–848.
[34] Ohnishi K, Shibata M, Murakami T. Motion control for advanced mechatronics. IEEE/ASME Trans Mechatronics. 1996;1(1):56–67.