A Neural Network Based Framework for Variable Impedance Skills Learning from Demonstrations

Yu Zhang[a,b], Long Cheng[a,b,*], Ran Cao[a,b], Houcheng Li[a,b], Chenguang Yang[c]

[a] *State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.*
[b] *School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China*
[c] *Bristol Robotics Laboratory, University of the West of England, Bristol BS16 1QY, U.K.*

Abstract

Robots are becoming standard collaborators not only in factories, hospitals, and offices, but also in people's homes, where they can play an important role in situations where a human cannot complete a task alone or needs the help of another person (i.e., collaborative tasks). Variable impedance control with contact forces is critical for robots to successfully perform such manipulation tasks, and robots should be equipped with adaptive capabilities because conditions vary significantly for different robotic tasks in dynamic environments. This can be achieved by learning human motion capabilities and variable impedance skills. In this paper, a neural-network-based framework for learning variable impedance skills is proposed. The proposed approach builds the full stiffness function with the acquired forces and position learned from demonstrations, and then is used together with the sensed data to achieve the variable impedance control. The proposed algorithm can adapt to unknown situations that change the learned motion skill as needed (e.g., adapt to intermediate via-points or avoid obstacles). The proposed framework consists of two parts: Learning motion features and learning impedance features. The motion features learning is validated by reproducing, generalizing, and adapting to transit points and avoiding obstacles in the LASA dataset. Impedance features learning is validated based on a virtual variable stiffness system that achieves higher accuracy (approximately 90%) compared to traditional methods in a manual dataset, and the whole framework is validated through a co-manipulation task between a person and the Franka Emika robot.

Keywords: Variable impedance skill, Learning from demonstrations, Skills learning, Human-robot interaction

1. Introduction

With the development of robotics, robots are widely used in various fields, such as manufacturing and rehabilitation [1, 2]. In these scenarios, robots are expected to perform complex manipulation tasks that require interaction with unknown and unstructured environments (i.e., humans), making fixed coding unfeasible. In this case, learning from demonstration (LfD) is an intuitive and user-friendly method that helps robots implicitly learn task constraints and acquire manipulation skills from demonstrations [3]. Most LfD methods mainly focus on learning motion trajectories; for robots operating in unknown and unstructured environments, however, learning motion trajectories alone is not sufficient. Extending robot learning capabilities to force and impedance domains [4, 5] is critical for robots working in dynamic environments.

In this paper, a neural network based framework is proposed for learning variable impedance skills (see Fig. 1). The learning framework encodes not only the motions but also the interaction model that encapsulates the expert dynamics. The main contributions of this paper are listed below:

- A dynamic system approach is proposed to learn the point-to-point motions using neural networks while maintaining

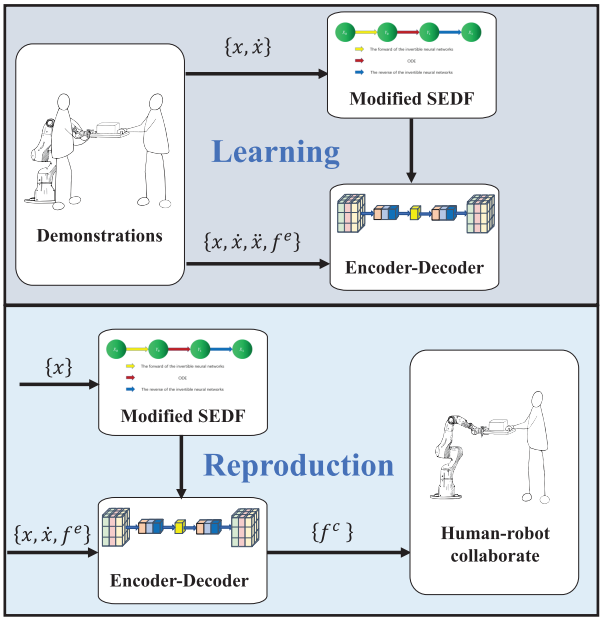


Figure 1: Structure of the proposed approach.

*Corresponding author
Email address: Long.cheng@ia.ac.cn (Long Cheng)

stability.

- The proposed algorithm can easily modify the generated motions to pass through via-points or avoid obstacles, while retaining the characteristic shape of the demonstration motions.
- A neural network with the encoder-decoder architecture is used to construct a function between stiffness and motion information and implement variable impedance control.

The paper is structured as follows: In Section 2, the related works are presented. In Section 3, the neural network which has the inherent property of keeping stability while modifying readily and the encoder-decoder structure which achieves the variable impedance control is introduced. In Section 4, the performance of the methods is evaluated by simulation and by experiments on the Franka Emika robot. In Section 5, the conclusion is made.

2. Related Work

The present work is mainly related to three areas, which are discussed in this section.

2.1. motion modeling

In the context of motion modeling, probabilistic approaches have achieved the desired performance in various scenarios [6, 7]. In [8], the motion features are extracted from multiple demonstrations using the Gaussian mixture model (GMM), and the new motions are generated using Gaussian mixture regression (GMR). Another powerful motion modeling tool is the dynamic system (DS), which is robust to perturbations [9, 10]. Dynamic Movement Primitives (DMP) is one of the most widely used DS algorithms that learns from a demonstration and generates motion trajectories based on a nonlinear spring-damper system [11], which has the inherent property of guaranteeing stability.

The stable estimator of dynamical systems (SEDS) is the first state-dependent DS algorithm proposed in [9]. It encodes the demonstration motions using the GMM and restricts the parameters of the GMR to guarantee global stability. However, the accuracy of motion modeling under strict stability constraints is low when learning from demonstrations, which is referred to as the *accuracy vs stability dilemma* [12].

Recently, several approaches have been proposed by researchers to improve the accuracy of reproduction with stability constraints. In [13], the use of a neural network to estimate the Lyapunov function based on demonstration is proposed, and the accuracy of the reproduction can be improved as the data of violation of the Lyapunov function is reduced. In [14], an algorithm for controlling Lyapunov function-based dynamics movements (CLF-DMs) is proposed which first learns a Lyapunov function that approximately matches the demonstration data, then uses a regression technique to model the motion, and finally guarantees the stability of the reproduction motions by solving an optimization problem online, which makes the algorithm complex and sensitive to the parameters. In [12], diffeomorphic transformation (τ-SEDS) is proposed to project the data into another space to improve the accuracy while keeping the learned motions stable. In [15], a fast stable learning method (FSM-DM) is proposed to model the demonstration motions using Extreme Learning Machine (ELM) considering stability, but the algorithm has unsatisfactory performance in reproducing and generating new motions. In [16], stable dynamical system learning using Euclideanizing flows (SDS-EF) is proposed to learn stable dynamical systems through a neural network to learn a diffeomorphism transformation which not only has good performance in generating new motions but also can gain high accuracy in reproduction, except for the cost of learning time. In [17], an attention-based approach for dynamical system to learn while keeping stability is proposed, which can be easily extended to learning high-dimensional complex motions.

The proposed algorithm also uses an invertible neural network like [16] that is stable and robust to perturbations, which does not need to calculate the Jacobian of the neural network.

2.2. motion modulation

After modeling motions learned from demonstrations, some new task conditions may be met in real experiments. Reinforcement learning (RL) is a solution to refine trajectories to meet the new conditions and achieve optimal performance. For instance, a variant of policy improvement with path integrals [18] is employed to optimize the parameters of the DMP. However, RL approaches that refine trajectories online as a time-consuming search for an optimal strategy are impractical. In contrast to the RL treatment, probabilistic movement primitives (ProMP) formulate the modulation of trajectories as a Gaussian conditioning problem, and therefore derive an analytical solution to adapt trajectories to new via-points or targets [19]. In [20], a kernelized movement primitive (KMP) is proposed that allows the robot to adapt to learned motor skills and to satisfy a variety of additional constraints that arise during the course of a task. In [21], a via-points Movement Primitive (VMP) is proposed that can adapt to arbitrary via-points by using a simple structured model to divide the trajectory into an elementary part and a modulation part and realize the modulation of the trajectory by adjusting them.

The proposed algorithm transfers this problem in a different space to a trajectory tracking problem, which is easy to implement while keeping more features of the original motion.

2.3. Variable impedance control

In real experiments, unstructured environments are constantly encountered, requiring variable impedance control. As humans have the ability to perform well in unstructured environments, some researchers have attempted to learn variable stiffness skills from human demonstrations [22]. In [23], the parameters of the variable admittance controller are self-regulated based on gait prediction by the Deep Gaussian Process method and the effectiveness is verified in both simulations and real experiments. In [24], an algorithm for estimating stiffness from

interaction force information in a human-robot cooperative task is developed. First, the demonstration trajectories are modeled as a task-parameterized Gaussian mixture model (GMM), and then the stiffness matrices are estimated using convex optimization in each Gaussian component. Finally, the estimated stiffness is used to implement a variable impedance controller. In [25] the stiffness is estimated using the measured interaction force, then the relationship between the stiffness matrices and the interaction forces is modeled by GMM, and GMR is used to reproduce the stiffness for given interaction forces. In [26], the impedance skills are learned from the depth images and then the desired control torques are output for the multi-fingered dexterous hand, which can realize the human-like compliant behavior by the end-to-end neural networks.

The proposed algorithm implements the variable impedance control using a neural network with an encoder-decoder structure that models the stiffness with both the interaction force and motion information.

3. Proposed Approach

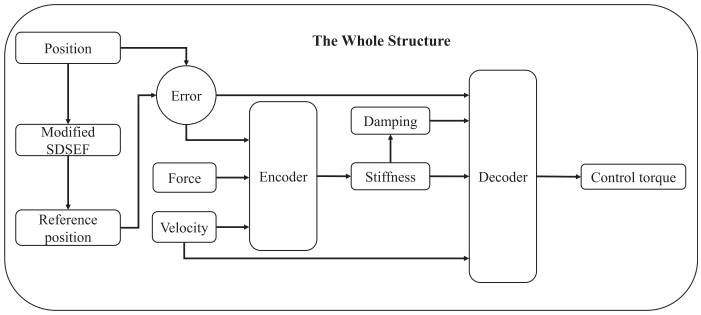


Figure 2: The structure of the whole framework to realize the variable impedance control.

In this section, a novel PbD architecture is proposed that combines two neural networks for learning and reproducing cooperative tasks. A modified stable dynamical system learning using euclideanizing flows (SDSEF) is used to learn the motions from the demonstration, and a neural network with encoder-decoder architecture is used to encode the motions with force data that encapsulates the dynamics of the demonstration behavior. The whole framework is presented in Fig. 2 and explained in the following sections.

3.1. Interaction Model

The motion of the robot's end-effector in the task space can be modeled as a single unit of mass on which the interaction forces $\boldsymbol{f}^e$ and the control input $\boldsymbol{f}^c$ act as follows:

$$\ddot{\boldsymbol{x}} = \boldsymbol{f}^e + \boldsymbol{f}^c, \tag{1}$$

where $\ddot{\boldsymbol{x}}$ is the acceleration of the movement and the simple dynamic model is borrowed from [27]. Therefore, setting the robot's motion control forces as $\boldsymbol{f}^c$ can achieve the desired task dynamics. To obtain the control forces, the robot motion during interaction in a virtual MSD system is modeled at each time step t as follows:

$$\ddot{\boldsymbol{x}}_t = \boldsymbol{K}_t^p(\boldsymbol{x}_t^r - \boldsymbol{x}_t) - \boldsymbol{K}_t^v\dot{\boldsymbol{x}}_t + \boldsymbol{f}_t^e, \tag{2}$$

where $\boldsymbol{K}_t^p$, $\boldsymbol{K}_t^v$ and $\boldsymbol{x}_t^r$ are the full stiffness matrix, the full damping matrix and the reference trajectory of the virtual system. $\boldsymbol{x}_t$, $\dot{\boldsymbol{x}}_t$, $\ddot{\boldsymbol{x}}_t$, and $\boldsymbol{f}_t^e$ are the position, velocity, acceleration, and interaction forces, respectively.

Notice that $\dot{\boldsymbol{x}}_t$ and $\ddot{\boldsymbol{x}}_t$ are the first and second time derivatives of $\boldsymbol{x}_t$, which can be acquired after the position and time have been obtained. Moreover, the interaction forces $\boldsymbol{f}_t^e$ are obtained by the force sensor mounted at the robot's end-effector. Varying both the stiffness and the attractor in this interaction model (2) can shape robot behavior as needed. To acquire an expert-like ideal behavior, the variables $\boldsymbol{x}_t^r$ and $\boldsymbol{K}_t^p$ will be learned from demonstrations provided by human teachers. These variables are learned using the framework in Fig. 2 that are explained in the rest of this section. Moreover, $\boldsymbol{K}_t^v$ can be specified according to the desired response of the interaction system (2) selected by experts or just chosen by the neural network learning from the demonstrations.

3.2. Motion learning and modulation

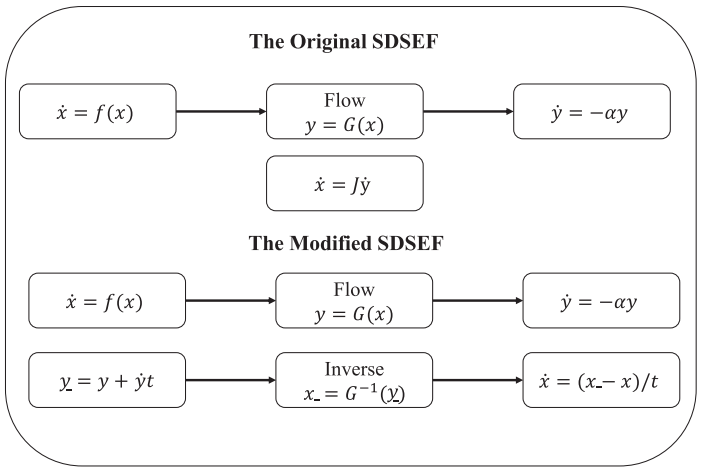


Figure 3: Comparison of the original SDSEF and the Modified SDSEF to model motions.

SDSEF [16] treats demonstration trajectories as goal-directed motions governed by a stable DS on a Riemannian manifold, which can be learned by invertible neural networks. The DS on the manifold is essentially the same as the DS in the original space, therefore, the generated trajectories are stable in the original space and converge to the goal point.

Considering that demonstration motions are modeled by a DS:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{R}^d, \tag{3}$$

where $\boldsymbol{x}$ is the state of motion, $\dot{\boldsymbol{x}}$ is the first-order time derivative of $\boldsymbol{x}$ and f provides the dynamics. After an initial state $\boldsymbol{x}_0$ is given, the whole motion is generated by solving the ordinary differential equation (ODE), which can be evaluated

with a numerical integrator (i.e., Runge-Kutta method and Euler method). When SDSEF is used to model the motions, the ODE can be transformed using invertible neural networks as follows:

$$\dot{\boldsymbol{y}} = -\alpha \boldsymbol{y},\ \forall \boldsymbol{y} \in \mathbb{R}^d, \tag{4}$$

where α is a positive constant, $\dot{\boldsymbol{y}}$ is the first-order time derivative of $\boldsymbol{y}$, $\boldsymbol{y} = G(\boldsymbol{x})$ and G is a diffeomorphism transformation achieved by the invertible neural networks. Therefore, the motions are generated by SDSEF using the Jacobian and numerical integrator methods.

In contrast to the original SDSEF, this work modifies the model as shown in Fig. 3. The motions are first generated on this Riemannian manifold and then transformed to the original space using the inverse of the trained invertible neural networks, so that it is no longer necessary to calculate the Jacobian of the forward pass of the invertible neural networks, and the motions can be explicitly expressed in Riemannian space utilizing the numerical integrator. With the given initial state, the whole motion on this Riemannian manifold can be generated as:

$$\boldsymbol{y} = G(\boldsymbol{x_0})e^{-\alpha t}. \tag{5}$$

Therefore, after training the model using the demonstration data, arbitrary motions can be obtained as $\boldsymbol{x} = G^{-1}(\boldsymbol{y})$ using the backward pass of the invertible neural networks. In this setting, the generated trajectories can be easily adapted to different starting or ending points and are robust to spatial perturbations.

Each coupling layer in SDSEF can be formally expressed as follows:

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{z}_k^a \\ \boldsymbol{z}_k^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_{k-1}^a \\ \boldsymbol{z}_{k-1}^b \odot \exp(s_k(\boldsymbol{z}_{k-1}^a)) + t_k(\boldsymbol{z}_{k-1}^a) \end{bmatrix}, \tag{6}$$

where $\exp$ denotes pointwise exponential and $\odot$ denotes pointwise product; $\boldsymbol{z}_k$ denotes the input or intermediate variable in the flow. The dimension of these parameters is $\boldsymbol{z}_k \in \mathbb{R}^{n\times 1}$, $\boldsymbol{z}_k^a \in \mathbb{R}^{n/2\times 1}$ or $\boldsymbol{z}_k^a \in \mathbb{R}^{(n+1)/2\times 1}$ and $\boldsymbol{z}_k^b \in \mathbb{R}^{n/2\times 1}$ or $\boldsymbol{z}_k^b \in \mathbb{R}^{(n-1)/2\times 1}$. For concise, assume that n is an even number. Just like [16], the $s_k(\cdot)$ is set as a neural network with the layer resembling an approximated kernel machine [28]. The Gaussian kernel is approximated by, which has the inherent property of imposing a strict smoothing constraint. The $s_k(\cdot)$ is expressed as:

$$s_k(\boldsymbol{z}) = \varphi(\boldsymbol{z})^{\mathrm{T}}\boldsymbol{w}, \tag{7}$$

where

$$\varphi(\boldsymbol{z}) = \sqrt{\frac{2}{m}} \begin{bmatrix} \cos(\boldsymbol{\alpha}_1^{\mathrm{T}}\boldsymbol{z} + \boldsymbol{\beta}_1) \\ \cos(\boldsymbol{\alpha}_2^{\mathrm{T}}\boldsymbol{z} + \boldsymbol{\beta}_2) \\ \vdots \\ \cos(\boldsymbol{\alpha}_m^{\mathrm{T}}\boldsymbol{z} + \boldsymbol{\beta}_m) \end{bmatrix} \in \mathbb{R}^{m\cdot[n/2]]\times[n/2]}, \tag{8}$$

and $\boldsymbol{w} \in \mathbb{R}^{1\times m\cdot[n/2]]}$ is the parameters learned by gradient descent. $\varphi(\boldsymbol{z})$ is an approximate for $K(\boldsymbol{z}, \boldsymbol{z'}) \approx \varphi(\boldsymbol{z})^{\mathrm{T}}\varphi(\boldsymbol{z'})$ which is composed of m randomly sampled Fourier features, the $K(\cdot)$ means the Gaussian kernel, the coefficients $\{\boldsymbol{\alpha}_i\}_{i=1}^m$ and the bias $\{\boldsymbol{\beta}_i\}_{i=1}^m$ are sampled from the zero-mean Gaussian distribution $\mathcal{N}(\boldsymbol{0}, l^{-2}\boldsymbol{I})$ and the uniform distribution $\mathrm{U}(0, 2\pi)$, respectively. l is a hyperparameter. The detail proof can be found in [28].

In the proposed algorithm, the $t_k(\cdot)$ is set as

$$t_k(\boldsymbol{z}) = \zeta(\boldsymbol{z})^{\mathrm{T}}\boldsymbol{w'}, \tag{9}$$

where

$$\zeta(\boldsymbol{z}) = \sqrt{\frac{2}{m}} \begin{bmatrix} \mathrm{sigmoid}(\boldsymbol{\gamma}_1^{\mathrm{T}}\boldsymbol{z}) \odot \boldsymbol{z} \\ \mathrm{sigmoid}(\boldsymbol{\gamma}_2^{\mathrm{T}}\boldsymbol{z}) \odot \boldsymbol{z} \\ \vdots \\ \mathrm{sigmoid}(\boldsymbol{\gamma}_m^{\mathrm{T}}\boldsymbol{z}) \odot \boldsymbol{z} \end{bmatrix} \in \mathbb{R}^{m\cdot[n/2]]\times[n/2]}, \tag{10}$$

and $\{\boldsymbol{\gamma}_i\}_{i=1}^m$ are also sampled from the zero-mean Gaussian distribution $\mathcal{N}(\boldsymbol{0}, l^{-2}\boldsymbol{I})$, $\mathrm{sigmoid}(\boldsymbol{\gamma z}) = 1/(1+\exp^{-\boldsymbol{\gamma z}})$ and $\mathrm{sigmoid}(\boldsymbol{\gamma z}) \odot \boldsymbol{z}$ can be seen as the Swish function in [29]. In this case, both the original trajectories and the transformed trajectories in the Riemannian manifold converge simultaneously to zero. It can be proved that $\boldsymbol{s}$ and $\boldsymbol{t}$ are continuous and continuously differentiable with respect to $\boldsymbol{z}$, which is essential for invertible neural networks [30].

In some cases, additional requirements can be met when generating the motion (i.e., some tasks require the robot to traverse certain specific locations in the workspace). It is not easy to modulate the trajectories generated by the dynamical system methods through the intermediate via-points, like the trajectories generated with ODE, which is the determination. To handle this problem, a variational ODE on the Riemannian manifold is proposed that allows the modulation of trajectories while preserving the motion features learned from demonstrations. When the via-points are provided, the corresponding points in the transformation space can be determined using the forward pass of invertible neural networks. Equation (4) indicates that these corresponding points are in the rectilinear trajectories in the transformation space. Therefore, to pass through particular points in the task space, the modulation of the trajectories can be achieved by following the rectilinear trajectory in the transformation space that contains the traversal point.

Given an initial point $\boldsymbol{x_0}$ and a via-point $\boldsymbol{x_v}$, the corresponding coordinates in the transformation space are $G(\boldsymbol{x_0})$ and $G(\boldsymbol{x_v})$. The initial motion in the transformation space is a rectilinear trajectory pointing to the zero point, which implies that the energy function

$$V = \frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} \tag{11}$$

is a dissipation that guarantees the stability of the proposed algorithm. Using the energy function (11), the initial energy and the via-point energy are obtained as $V_0 = \frac{1}{2}\boldsymbol{G(x_0)}^{\mathrm{T}}\boldsymbol{G(x_0)}$ and $V_v = \frac{1}{2}\boldsymbol{G(x_v)}^{\mathrm{T}}\boldsymbol{G(x_v)}$, respectively. The point with the same energy as V_v in a rectilinear path containing $G(\boldsymbol{x_v})$ can be easily determined as $\boldsymbol{y}_s$. Then the reference point on the desired trajectory can be selected as follows:

$$\boldsymbol{y}_r = \boldsymbol{y}_s - \alpha\beta\boldsymbol{y}_s, \tag{12}$$

where β is a positive constant selected by experience, if V_0 is close to V_v, β can be selected little and vice versa. Therefore,

the new ODE in the transformed space can be modified as:

$$\dot{\boldsymbol{y}} = \alpha \frac{|\boldsymbol{y}|}{|\boldsymbol{y}_r - \boldsymbol{y}|}(\boldsymbol{y}_r - \boldsymbol{y}),\ \forall \boldsymbol{y} \in \mathbb{R}^d, \tag{13}$$

where $|\cdot|$ denotes the 2-norm. Finally, using the new ODE in (13) in the transformation space, the trajectories can be generated to pass through the intermediate via-point, while preserving the motion feature learned from the demonstrations.

When there is more than one passing point, the via-points have to be sorted according to the energy calculated in (11) and the reference point to be tracked must change when the generated motions pass through a via-point, therefore, the entire generated trajectories can pass through one via-point after another until they converge to the target point. It is worth mentioning that the modulation of trajectories with the proposed method does not require retraining of the model, which means that the proposed algorithm does not increase the computation time to modulate the generated trajectories to pass through the via-points.

In practise, there are often static obstacles that require trajectories to be modulated to avoid them. Since the proposed algorithm can pass through an arbitrarily via-point near the demonstration region, a method for modulating the trajectory to avoid an obstacle $\boldsymbol{x_o}$ is proposed that assumes the trajectory can pass a point in the security boundary while always maintaining a safe distance from the obstacle. The security boundary is defined as a circle with a radius r to simplify the problem and the generated trajectory would be tangent to the circle, which means that the straight-line trajectory in the transformation space is also tangent to the boundary transformed by the circle. The tangent points $\boldsymbol{x_p}$ are calculated as follows:

$$\begin{cases} |\boldsymbol{x_p} - \boldsymbol{x_o}| = r, \\ < G^{-1}(\gamma G(\boldsymbol{x_p})) - \boldsymbol{x_p}, \boldsymbol{x_p} - \boldsymbol{x_o} >= 0, \end{cases} \tag{14}$$

where γ is a constant selected by experience (approximately equal to 1), $G(\cdot)$ is the diffeomorphism transformation realized by the trained SDSEF, $|\cdot|$ denotes the 2-norm and $< \cdot >$ means the inner product. The complete modulation for generating trajectories that avoid an obstacle in a 2D coordinate system is summarized in Algorithm 1, which can be easily extended to high-dimensional space.

Algorithm 1: Generate a trajectory to avoid obstacle

Input: initial point x_0, obstacle o, safety radius r.
Output: a generated trajectory to avoid obstacle while keeping the characteristics of demonstrations.

1: generate a trajectory from the initial point using the modified SDSEF
2: **if** the trajectory is always keeping a safe distance **then**
3: output the trajectory
4: **else**
5: Calculate tangent point x_{pn} in the security boundary which is nearer to the generated trajectory. Using x_0 as the initial point and x_{pn} as the via-point to generate a new trajectory.
6: **end if**

3.3. Stiffness Estimation Model

The purpose of estimating the stiffness matrix is to shape the compliance of the robot like that of the human teacher. It can be learned from equation 1 and equation 2 that estimating the stiffness matrix is not the goal but an intermediate process to obtain the control torque for the robot. Unlike the method proposed in [24, 25], which first estimates the stiffness matrix from the demonstrations using convex optimization or least square method, then builds the model between the stiffness and Gaussian component or force data, and finally achieves variable impedance control using the built model and interaction model, an encoder and decoder neural network consisting of a modified SDSEF is proposed to achieve variable impedance control.

The demonstration data are recorded when the human teacher executing the interaction tasks repetitively as $\left\{\boldsymbol{x}_{k,n}, \dot{\boldsymbol{x}}_{k,n}, \ddot{\boldsymbol{x}}_{k,n}, \boldsymbol{f}^e_{k,n} \mid k = 1, 2, ..., K_n;\ n = 1, 2, ..., N_d\right\}$, where n is the nth demonstration and k is the kth sampling time; N_d and K_n denote the total number of demonstrates and the whole sampling number of the nth demonstrated motion, respectively. The state variable $\boldsymbol{x}$ represents the coordinates of the robot's end-effector in Cartesian space and $\boldsymbol{f}^e$ are the forces read by the force sensor mounted at the robot's end-effector. There is no stiffness in the demonstrations and it is tough to obtain the ground-truth stiffness because the actual reference motions are unknown. However, the estimated reference motions can be generated using the modified SDSEF method described in 3.2, and the interaction model (2) can be used to estimate the varied stiffness.

The structure of the proposed algorithm is shown in Fig. 2. The encoder aims to learn a lower triangular matrix, which is the Cholesky decomposition of the stiffness after sensing position, velocity, and force, to satisfy the symmetric and positive definite (SPD) constraints, while the decoder expresses the interaction model (2) that uses the estimated stiffness and sensor data to output the torque of the robot. The encoder is designed as a three-layer network, which can be viewed as a function $\boldsymbol{K}^p_t = F(\boldsymbol{x}_t, \dot{\boldsymbol{x}}_t, \boldsymbol{f}^e_t)$ learned from demonstrations. Therefore, just like human teachers, the learned model can change compliance after perceiving this information.

In the framework proposed in this paper, the neural network directly outputs the control torque $\boldsymbol{f}^c$ in (1) for the robot after acquiring the robot's perception and haptic information from the force sensors, which is then transformed to joint torques $\boldsymbol{\tau}$ of the robot using the Jacobian transpose J^{T} as follows: $\boldsymbol{\tau} = J^{\mathrm{T}}\boldsymbol{f}^c$. It is worth noting that when using the encoder and decoder neural network, the damping term in the interaction model (2) can be simply specified as $\boldsymbol{K}^v_t = \boldsymbol{T}^{\mathrm{T}}_t(\zeta \boldsymbol{\Lambda}^{\frac{1}{2}}_{\boldsymbol{t}})\boldsymbol{T}_t$, where $\boldsymbol{T}_t$ and $\boldsymbol{\Lambda}$ are the eigenvector and eigenvalue of $\boldsymbol{K}^p_t$ as $\boldsymbol{K}^p_t = \boldsymbol{T}^{\mathrm{T}}_t \boldsymbol{\Lambda_t} \boldsymbol{T}_t$, and $\zeta \in \mathbb{R}^+$ is a manually specified tuning parameter, which is difficult to achieve using the least square methods in [25]. In contrast, the eigenvalue decomposition of $\boldsymbol{K}^p_t$ can be easily implemented using the package in pytorch (i.e., torch.linalg.eigh).

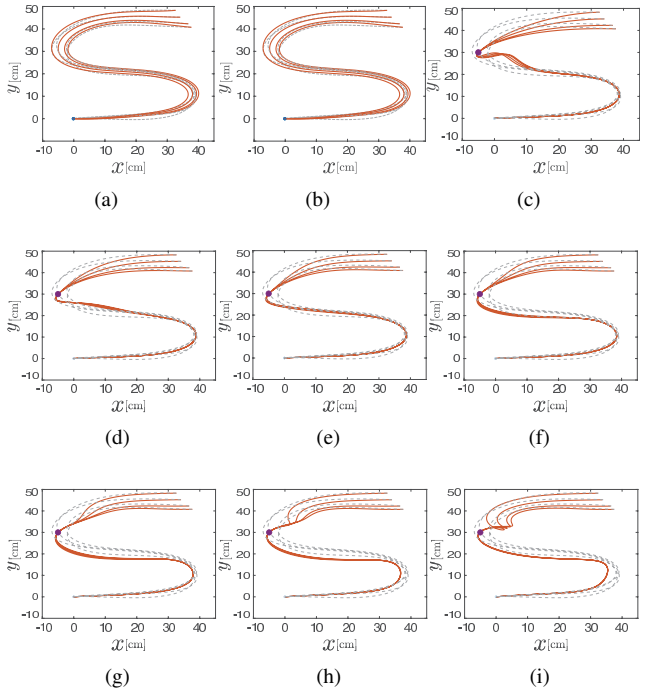


Figure 4: Results obtained by applying KMP to both the reproducing and modulating trajectories. (a) shows the reproducing trajectories with the same starting points while the others are the modulating trajectories with different predetermined times for passing through the via-point.

4. Experiment Results and Discussions

To evaluate the effectiveness of the proposed neural network based framework, both simulations and real experiments are conducted. The simulation is implemented in Python 3.8 and the experiment is implemented using Franka Emika robot in C++ 14 on the platform of CPU Inter Core i5-8300H, Ubuntu 20.04. The experiments used in the evaluation are summarized as follows:

- The effect of the modified SESEF in modulating the trajectories is verified by simulation in the LASA dataset [31] to perform the motion learning and modulation of the trajectories.
- The performance of the estimated stiffness is evaluated in a self-made dataset generated by a 2-DoF MSD with a manually designed $\boldsymbol{f}_t^e$ and $\boldsymbol{K}_t^p$ with the attractor of the interaction model (2) at zero.
- The whole framework is evaluated using a common human-robot transportation task by a human with the Franka Emika robot.

4.1. Trajectories Modulation in Simulation

In this section, several handwritten letter trajectories are used as demonstrations to train the modified SDSEF, various points near the demonstrate domain are provided as passing points, and then the proposed algorithm is used to modulate the generated trajectories to pass these points. To evaluate the effectiveness of the proposed algorithm, the KMP [20] which can realize trajectory modulations and extrapolations is also implemented for comparison purposes. Finally, several points with a certain radius are provided as obstacles to show the performance of the proposed algorithm in avoiding obstacles.

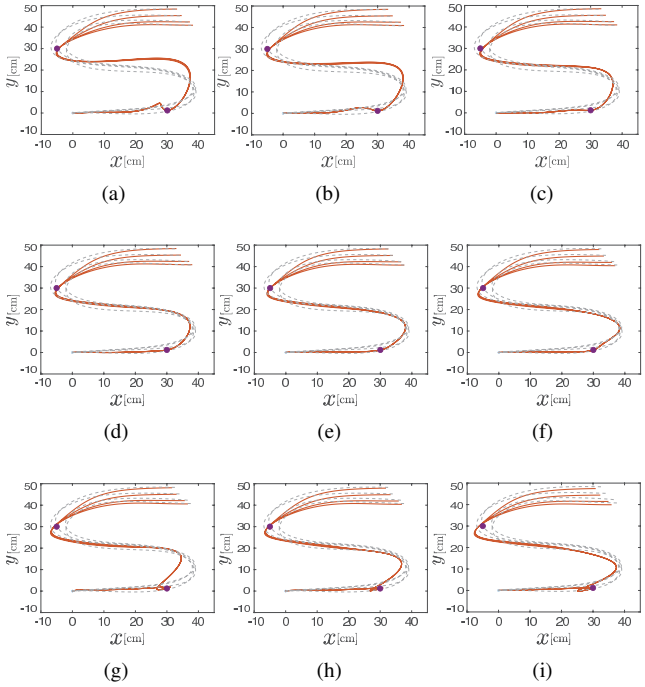


Figure 5: Results are obtained by applying KMP for passing two via-points, where the time for passing the first point is $0.6s$. The time for passing the second point is set from $1.45s$ to $1.95s$ in photos a to f, and from $1.75s$ to $1.95s$ in frames g to i with unreasonable velocity.

KMP is an effective method for modeling motion with time inputs. Fig. 4 displays different trajectory modulation using KMP with different predetermined times to pass the via-point. Fig. 4(a) shows the reproduction trajectories without modulation with the same starting points as the demonstrations. The dotted gray curves are the demonstration trajectories and the solid pink curves are the generated trajectories with the same starting points. The relevant parameters of KMP are the same as 4. The time duration of each trajectory is $2s$ and Fig. 4(b) to Fig. 4(i) are the modulation trajectories with the predetermined times for passing the via-point at $0.3s$ to $1.0s$ with interval $0.1s$.

It can be seen from Fig. 4 that the KMP can acquire a good performance in the reproduction and modulation trajectories when a suitable time is chosen to pass the points. However, setting a specified time for passing the via-points is not convenient in practical, and an inappropriate time leads to poor trajectories generation (i.e., Fig. 4(b) and Fig. 4(i)).

Fig. 5(a) to Fig. 5(f) display different trajectory modulation passing two via-points using KMP, where the time for passing the first point is $0.6s$, and the time for passing the second point is $1.45s$ to $1.95s$ with interval $0.1s$. Fig. 5(g) to Fig. 5(i) show different modulations of trajectories passing two via-points with the same time as in Fig. 5(d) to Fig. 5(f), but with an inappropriate velocity setting.

Fig. 6 displays different trajectory modulation using the

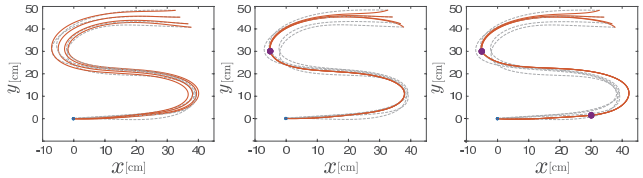


Figure 6: Results obtained by applying the proposed algorithm on both reproducing and modulating trajectories.

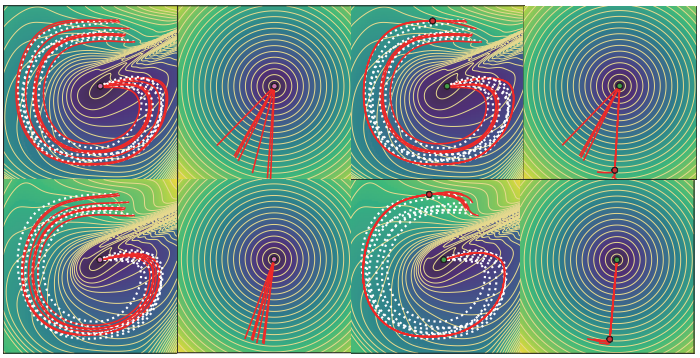


Figure 7: Heatmap results of the proposed algorithm to learn the motion of the handwriting letter "G". The first and second raw images are generated without and with energy constraints, respectively.

modified SDSEF algorithm. The number of the coupling layers is set as 8, the neuron is set as 128, learning rate is set as 0.001 and the optimizer is set to "AdamW" from [32].

It can be observed from Fig. 5 and Fig. 6 that both KMP and the proposed algorithm successfully generate trajectories that pass through the via-point. However, the trajectories generated by the proposed algorithm can preserve more features of the demonstration data that can be quantitatively measured by the position smoothness c_p, which is defined as $c_p = \frac{1}{K_n}\sum_{k=1}^{K_n-1}(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k)^{\mathrm{T}}(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k)$. The result of c_p of different trajectories passing through one via-point generated by KMP and the proposed algorithm is summarized in Table 1. The trajectories generated by KMP use the predetermined time of $0.6s$ to pass through these via-points. It can be observed from Table 1 that the c_p between the proposed algorithm and the demonstrations are very close, which means that the proposed algorithm can retain more features of the demonstrations. In particular, trajectories modulated by the proposed algorithm do not require a well-designed time and speed to pass the desired point.

It is worth mentioning that the performance of the proposed algorithm in modulating trajectories is extremely dependent on the value calculated from the energy function (11) on the Riemannian manifold. However, after learning, the rectilinear trajectories in the transformation space are uncontrolled, which means that the energy of the initial points can vary greatly, so the proposed algorithm cannot pass through some via points near the demonstration zone.

A loss function ensures that the energy of the starting points is close to each other is very important for the modulation of

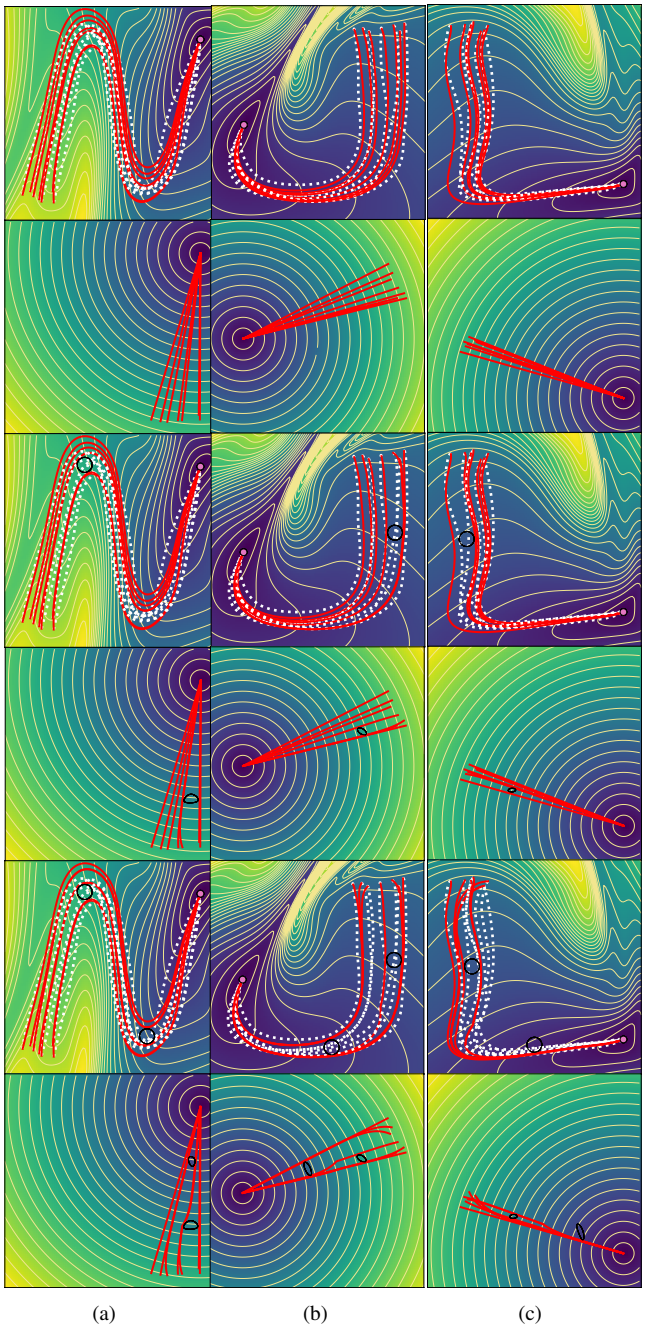


Figure 8: Heatmap results for three handwriting letters are obtained by applying the proposed algorithm in reproducing trajectories and avoiding obstacles. From top to bottom, the generated trajectories in the original space with the same starting points as the demonstrations, the generated trajectories in the transformation space, and the generated trajectories in the original space avoiding one obstacle, the corresponding trajectories in the transformation space, the generated trajectories in the original space avoiding two obstacles, and the corresponding trajectories in the transformation space, respectively.

Table 1: The values of trajectories smoothness

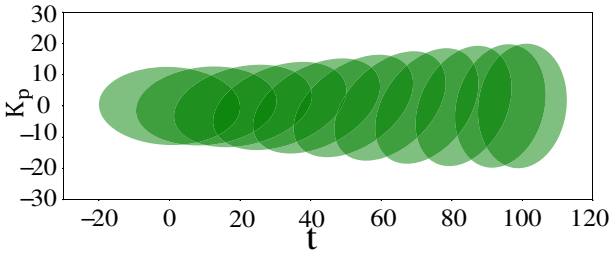
	c_p of KMP	c_p of the proposed algorithm	c_p of the demonstration
First Trajectory	0.097	0.3516	0.3725
Second Trajectory	0.0966	0.3560	0.3606
Third Trajectory	0.0957	0.3624	0.4095
Fourth Trajectory	0.0977	0.3405	0.3999
Mean	0.0967	0.3526	0.3857

the trajectories and the results in learning the handwritten letter "G" are shown in Fig. 7. The first raw images show the trajectories generated by the model without considering the energy constraints, while the second raw photos show the trajectories generated considering the energy constraints. In each raw pictures are shown from left to right as, the movements which arise in the original space, in the transformation space, when trying to pass through a via-point with comparatively high energy in the original space and in the transformation space. In Fig. 7, the white dotted lines represent the demonstration data, the red solid lines demonstrate the generated trajectories from the same starting points with demonstrations, and the red dot is the via-point. The reproduction trajectories are more similar to the demonstrations when the model is trained without considering the energy constraints which are calculated by the swept error area (SEA) error [14] but are invalid for some starting points to pass through some points near the demonstration region. To overcome this dilemma, more parameters may be required in the model (i.e., the number of coupling layers is increased).

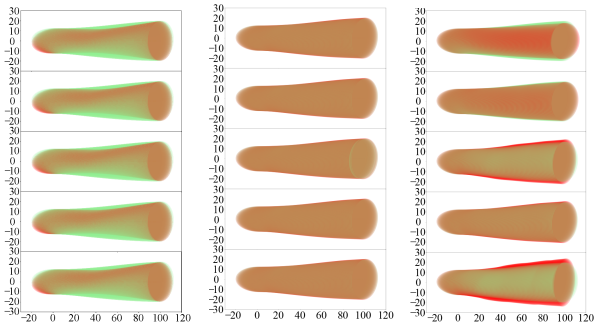
In real experiments, there are often obstacles. Fig. 8 shows the heatmap of three handwritten letter trajectories reproduction and obstacles avoided using the proposed algorithm in the LASA dataset. From top to bottom are shown the generated trajectories in the original space with the same starting points as in the demonstrations, the generated trajectories in the transformation space, the generated trajectories in the original space to avoid one obstacle, the corresponding trajectories in the transformation space, the generated trajectories in the original space to avoid two obstacles, and the corresponding trajectories in the transformation space, respectively. Moreover, the white dotted lines represent the demonstration data, the red solid lines show the generated trajectories from the same starting points with the demonstration, the black solid lines are the security boundary to avoid obstacles, and the yellow solid lines are the contours obtained from the Lyapunov-function $V(\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}$. It can be seen that the proposed algorithm can efficiently handle the obstacles avoidance problems while preserving the main features of the demonstrations.

4.2. *Stiffness Estimation in Simulation*

Before the stiffness is estimated in the simulation experiment, manually designed f^e profiles are used to calculate the $[x_t, \dot{x}_t, \ddot{x}_t]$ of the 2-DoF MSD system at each step t, where the specific stiffness matrix varies in a rule. The manually designed



(a)



(b) (c) (d)

Figure 9: Results of applying the compared algorithm and the proposed algorithm in estimating the stiffness. Green demonstrates the truth stiffness, which changes during the process and red is the estimated stiffness. If the estimated stiffness ellipse matches the true stiffness ellipse, it turns brown.

Table 2: The mean square error of the control torque in two dimensions

	The proposed algorithm in [25]	Convex Optimization plus the encoder network	The encoder-decoder networks
First Trajectory	2.97 / 1.06	0.03 / 0.04	0.06 / 0.10
Second Trajectory	0.46 / 0.21	0.04 / 0.06	0.07 / 0.13
Third Trajectory	5.97 / 3.16	0.11 / 0.13	0.09 / 0.18
Fourth Trajectory	0.52 / 0.23	0.06 / 0.08	0.12 / 0.24
Fifth Trajectory	7.12 / 4.99	0.08 / 0.10	0.15 / 0.31
Mean	3.40 / 1.93	0.06 / 0.08	0.10 / 0.19

stiffness is considered as the ground truth to compare the performance of the estimation with the state.

Ten groups of different but proportional interaction forces are applied to the 2-DoF MSD system to generate ten trajectories. The stiffness is initialized starting from a horizontally oriented ellipsoid and then continuously rotated clockwise $\boldsymbol{R}_t^{\mathrm{T}}\boldsymbol{K}^p\boldsymbol{R}_t$ until it is vertically oriented (as shown in Fig. 9(a)). Five trajectories with corresponding forces are set as the training dataset while the rest are used as the test dataset. The encoder network of the proposed algorithm is a three-layer network with 32 hidden neurons and 3 output neurons whose activation functions are relu. The compared algorithm is proposed in [25], which leads to the fact that the stiffness estimates obtained by convex optimization can reach a higher accuracy. For a fair comparison, the stiffness estimation of the compared algorithm is performed using convex optimization which can be realized by CVXPY (a software package in python for convex optimization), and the damping is set as a constant matrix with the same

coefficient as the compared algorithm in [25]. Since the stiffness estimated from convex optimization is close to the ground truth, the estimated stiffness can be used as a label to train the encoder network.

The estimated stiffness profile and the actual stiffness profile for the five test trajectories are displayed in Fig. 9(b), Fig. 9(c), and Fig. 9(d). The stiffness matrices are shown as stiffness ellipses which were proposed in [33] for graphical visualization of stiffness matrix. Intuitionally, the convex optimization together with the encoder network can achieve the best performance in estimating the stiffness profile of the test data. The algorithm proposed in [25] performs well when the test data is very close to the training data and the encoder-decoder network does not perform poorly on all test data.

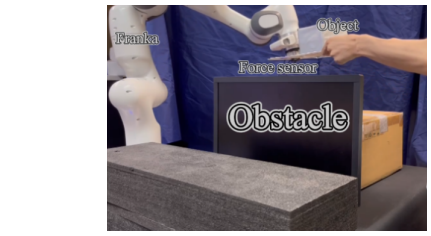
As mentioned in 3.3, the purpose of stiffness estimation is to determine the control torque $\boldsymbol{f}^c = -\boldsymbol{K}_t^p \boldsymbol{x}_t - \boldsymbol{K}_t^v \dot{\boldsymbol{x}}_t$. For the real experiment demand, the calculated control torque is used for comparison, and the results are displayed in Table 2. It is obvious that the convex optimization plus the encoder network can still achieve the best accuracy in comparing the truth value in both dimensions, while the encoder-decoder networks can obtain the approximate accuracy. Meanwhile, the algorithm proposed in [25] would get poor performance when the trajectory is away from the demonstrations. The main reason for the poor performance maybe be that only modeling the stiffness with the interaction force, which ignores the relationship between the stiffness and the motion. The deficiency is circumvented by the proposed algorithm.

However, convex optimization assumes that the stiffness profile is totally the same on the different trajectories at the same time, which does not correspond to the facts in real experiments. Therefore, in the real robot experiment, the encoder-decoder is applied to output the torque to realize the variable impedance control.

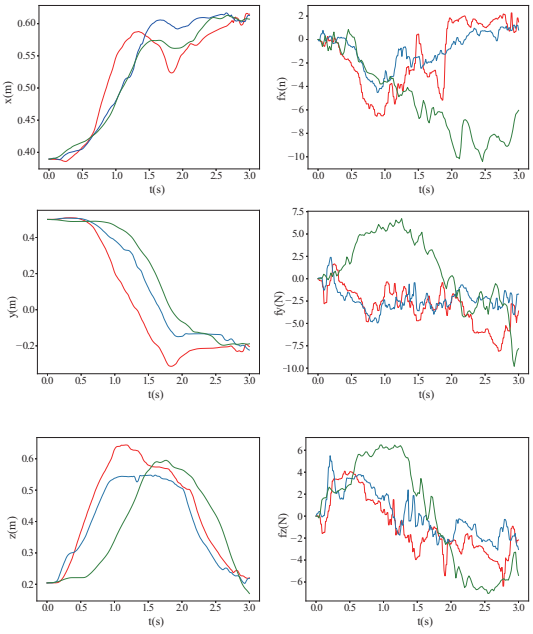
4.3. Validation on Robot

The transport task aims to teach a robot to transport an object cooperatively with a person (see Fig. 10). Initially, one participant contacts the object while the other person holds the robot, which operates in a zero gravity model to transport the object to the target location, which is shown as in Fig. 1. Motion information is recorded by the robot, while force data is acquired by the three-axis force/torque sensor equipped on the end-effector. Although both the position and orientation may vary in the demonstrations over several repetitions, only the position information is concerned in this paper to simplify the experiment.

It is worth pointing out that the algorithm in this experiment models the motions with time independence, so the human does not have to perform the same in every demonstration that is more user-friendliness. In this experiment, ten demonstration data are recorded to train the modified SDSEF model and the encoder-decoder neural networks. After learning, the trained models are used to control the robot to test the reproduction and generalization performance of the transportation task. Preset maximum torques calculated from the demonstrations are used to compare the torques output of the model to guarantee that the real control torques are in the safe range.



(a)



(b)

Figure 11: Results are shown for the three-dimensional position and force information in the task, obtained by applying the proposed algorithm compared to the zero gravity model, with and without obstacle avoidance considered. The red solid line expresses the results without considering obstacle avoidance, the blue solid line shows the results considering obstacle avoidance while the green solid line means the results acquired when the robot operates in zero gravity mode.

Some snapshots of the experiment are presented in Fig. 10. At the beginning of the first row, when the person does not apply any force to move the object, the robot is in an approximate zero gravity model as the calculated stiffness is small and the output control torques are also small, which cannot drive the robot to move. Then if the human applies a force to transpose the object, the robot moves and the calculated stiffness varies, causing the output control torques to change and the robot begins to cooperate with the human to accomplish the task. In the second row of Fig. 10, the person applies a wrong force and tries to move away from the target point, as the person and the robot transpose the object together, the robot will try to move the object to the target point, and if the person does not insist on the incorrect style, they will transpose the object to the target point. By the figures on the third row of Fig. 10, it can be seen that when the start and target points are changed, the pro-

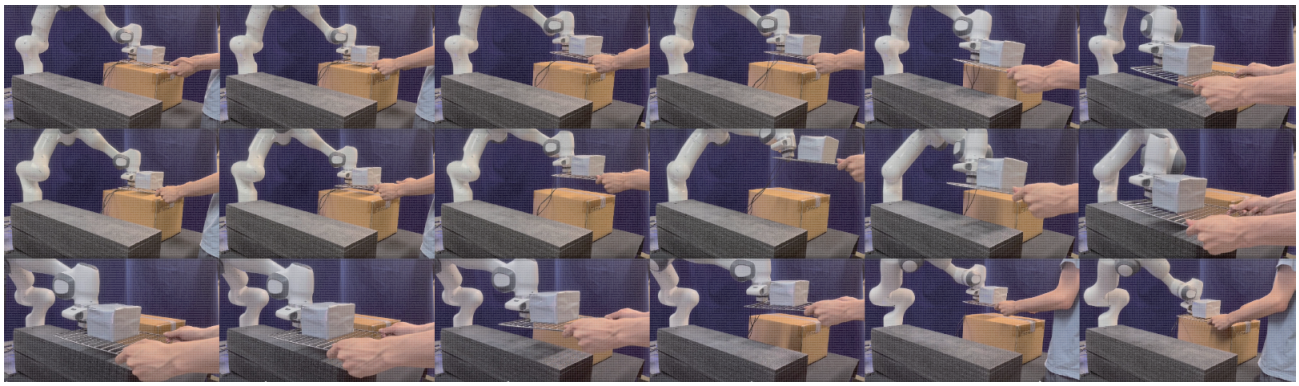


Figure 10: Snapshots of execution of a transportation task with a person and the Franka Emika robot in three-dimensional Cartesian space. The first raw photos illustrate the person and the robot transporting the object from a starting point to the target point with the person maintaining the motion style as in the demonstrations; the second raw photos show the person using a different style in the cooperation task and trying to deviate from the target point; the third raw photos present the robot and the person starting from a different initial point to a different goal point.

posed algorithm can guarantee that the robot cooperates with the person to complete the task successfully.

To evaluate the effectiveness of the proposed algorithm in avoiding an obstacle with a human while transporting the object. Three tests are performed with different configurations: 1) the robot works in zero gravity model, 2) the robot works without considering obstacle avoidance, 3) the robot works considering obstacle avoidance. The recorded information of the three tests is displayed in Fig. 11(b) in green, red and blue colors, respectively. The pictures in the left column in Fig. 11(b) show the position information of the end effector measured by the robot in three dimensions, where the horizontal axis is time. The photos in the left column display the consistent interaction force detected by the force sensor in the three dimensions.

When the robot works in zero gravity model, it can be dragged anywhere in the working area, and the object is completely transposed by the human from the starting point to the destination point. When it works without considering obstacle avoidance, the generated reference trajectory hit the obstacle in the upper half of the transportation task, and the human must apply a force to avoid the obstacle. This means that in the first half of the task, the human dominates the task and when it avoids the obstacle, the modified SESEF generates a new reference trajectory, and the human and the robot cooperate to finish the task. When the robot works considering the avoidance of the obstacle, the human and the robot cooperate to complete the task from the starting point to the target point, which means that the energy integrated by the interaction force provided by the human is lowest during this process. The calculated energies in the three tests are $0.395N \cdot s$, $0.214N \cdot s$, and $0.151N \cdot s$, respectively. This proves that the proposed algorithm considering obstacle avoidance is the most relaxed method for the human to cooperate with the robot to complete the task, and that the zero gravity model is the most laborious method for the human to complete the same task.

The two experiments confirm the effectiveness of the proposed neural network based framework in learning impedance skills from demonstrations that can not only work well in the regular pattern to transport an object but also work well to transport an object while avoiding an obstacle.

5. Conclusions and Future Work

In this paper, a neural network framework is proposed for variable impedance control learning from demonstration. The performance of the proposed framework is investigated using the demonstrated examples in the LASA handwriting dataset and a manually designed dateset. The experimental results confirm the effectiveness of the proposed method.

It should be noted that the absolute stability of the variable impedance control is not studied and the hyperparameters of the proposed algorithm are designed by the trial-and-error method which can be studied and optimized in future works. Meanwhile, the training process of the model is time-consuming compared to the state-of-the-art methods, which can also be improved.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (Grants 62025307 and U1913209) and the Beijing Natural Science Foundation (Grant JQ19020).

[1] H. Li, L. Cheng, Z. Li, and W. Xue, "Active disturbance rejection control for a fluid-driven hand rehabilitation device," *IEEE/ASME Transactions on Mechatronics,* vol. 26, no. 2, pp. 841-853, 2021.

[2] Y. Ma, D. Xu, and F. Qin, "Efficient insertion control for precision assembly based on demonstration learning and reinforcement learning," *IEEE Transactions on Industrial Informatics,* vol. 17, no. 7, pp. 4492-4502, 2021.

[3] H. Ravichandar, A.S. Polydoros, and S. Chernova, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control Robotics and Autonomous Systems,* vol. 3, no. 1, pp. 297-330, 2020.

[4] C. Zeng, Y. Li, J. Guo, Z. Huang, N. Wang and C. Yang, "A Unified Parametric Representation for Robotic Compliant Skills With Adaptation of Impedance and Force," *IEEE/ASME Transactions on Mechatronics,* vol. 27, no. 2, pp. 623-633, 2022.

[5] X. Yu, P. Liu, W. He, Y. Liu, Q. Chen and L. Ding, "Human-Robot Variable Impedance Skills Transfer Learning Based on Dynamic Movement Primitives," *IEEE Robotics and Automation Letters,* vol. 7, no. 3, pp. 6463-6470, 2022.

[6] A. Paraschos, C. Daniel, J. Peters and G. Neumann, " Using Probabilistic Movement Primitives in Robotics," *Autonomous Robots,* vol. 42, no. 3, pp. 1573-7527, 2018.

[7] J. Dong, M. Mukadam, F. Dellaert and B. Boots, "Motion Planning as Probabilistic Inference Using Gaussian Processes And Factor Graphs" *in Proceedings of Robotics: Science and Systems*, Michigan, USA, June, 2016, pp. 1-9.

[8] S. Calinon, F. Guenter and A. Billard, "On Learning, Representing, and Generalizing a Task in a Humanoid Robot," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 37, no. 2, pp. 286-298, 2007.

[9] S.M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Transactions on Robotics,* vol. 27, no. 5, pp. 943-957, 2011.

[10] M. Saveriano and D. Lee, "Distance based dynamical system modulation for reactive avoidance of moving obstacles," *in Proceedings of International Conference on Robotics and Automation,* Hong Kong, China, May, 2014, pp. 5618-5623.

[11] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation,* vol. 25, no. 2, pp. 328-373, 2013.

[12] K. Neumann and J.J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robotics and Autonomous Systems,* vol. 70, pp. 1-15, 2015.

[13] K. Neumann, A. Lemme, and J.J. Steil, "Neural learning of stable dynamical systems based on data-driven Lyapunov candidates," *in Proceedings of IEEE International Conference on Intelligent Robots and Systems,* Tokyo, Japan, Nov, 2013, pp. 1216-1222.

[14] S.M. Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems,* vol. 62, no. 6, pp. 752-765, 2014.

[15] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin and C. Xu, "Fast and stable learning of dynamical systems based on extreme learning machine," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 49, no. 6, pp. 1175-1185, 2019.

[16] M.A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," *in Proceedings of the 2nd Conference on Learning for Dynamics and Control*, Online, June, 2020, pp. 630-639.

[17] Y. Zhang, L. Cheng, H. Li and R. Cao, "Learning Accurate and Stable Point-to-Point Motions: A Dynamic System Approach," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1510-1517, 2022.

[18] B. Jonas, S. Freek, T. Evangelos, and S. Stefan, "Learning variable impedance control," *The International Journal of Robotics Research,* vol. 30, no. 7, pp. 820-833, 2011.

[19] A. Paraschos, C. Daniel, J.R. Peters, and G. Neumann, "Probabilistic movement primitives," *in Proceedings of the Advances in Neural Information Processing Systems,* Lake Tahoe, U.S.A, Dec., 2013, pp. 2616-2624.

[20] Y.L. Huang, L. Rozo, J. Silverio and D. Caldwell, "Kernelized Movement Primitives," *The International Journal of Robotics Research,* vol. 38, no. 5, pp. 833-852, 2019.

[21] Y. Zhou, J. Gao and T. Asfour, "Learning Via-Point Movement Primitives with Inter- and Extrapolation Capabilities," *in Proceedings of International Conference on Intelligent Robots and Systems,* Macau, China, 2019, pp. 4301-4308.

[22] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters,* vol. 5, no. 4, pp. 6129-6136, 2020.

[23] Z. Chen, Q. Guo, T. Li, Y. Yan and D. Jiang, "Gait Prediction and Variable Admittance Control for Lower Limb Exoskeleton With Measurement Delay and Extended-State-Observer," *IEEE Transactions on Neural Networks and Learning Systems,* Early Access.

[24] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras, "Learning physical collaborative robot behaviors from human demonstrations," *IEEE Transactions on Robotics,* vol. 32, no. 3, pp. 513-527, 2016

[25] F.J. Abu-Dakka, L. Rozo, and D. Caldwell, "Force-based variable impedance learning for robotic manipulation," *Robotics and Autonomous Systems,* vol. 109, pp. 156-167, 2018.

[26] C. Zeng, S. Li, Z. Chen, C. Yang, F. Sun and J. Zhang, "Multifingered Robot Hand Compliant Manipulation Based on Vision-Based Demonstration and Adaptive Force Control," *IEEE Transactions on Neural Networks and Learning Systems,* Early Access.

[27] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation,* vol. 3, no. 1, pp. 43-53, 1987.

[28] R. Ali and R. Benjamin, "Random features for large-scale kernel machines," *in Proceedings of neural information processing systems,* Hyatt Regency, Canada, 2008, pp 1177-1184.

[29] P. Ramachandran, B. Zoph, and Q.V. Le, "Searching for activation functions," arXiv preprint arXiv:1710.05941, 2018.

[30] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *in Proceedings of International Conference on Learning Representations,* Toulon, France, April, 2017, pp. 1-32.

[31] Available: https://bitbucket.org/khansari/lasahandwritingdataset/.

[32] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of International Conference on Learning Representations*, New Orleans, United States, May 2019, pp. 1–8.

[33] F.A. Mussa-Ivaldi, N. Hogan, and E. Bizzi, "Neural, mechanical, and geometric factors subserving arm posture in humans," *The Journal of Neuroscience*, vol. 5, no. 10, pp. 2732-2743, 1987.