Robot Learning System Based on Adaptive Neural Control and Dynamic Movement Primitives

Chenguang Yang, *Senior Member, IEEE*, Chuize Chen, Wei He, *Senior Member, IEEE*, Rongxin Cui, *Member, IEEE*, and Zhijun Li, *Senior Member, IEEE*

***Abstract*—This paper proposes an enhanced robot skill learning system considering both motion generation and trajectory tracking. During robot learning demonstrations, dynamic movement primitives (DMPs) are used to model robotic motion. Each DMP consists of a set of dynamic systems that enhances the stability of the generated motion toward the goal. A Gaussian mixture model and Gaussian mixture regression are integrated to improve the learning performance of the DMP, such that more features of the skill can be extracted from multiple demonstrations. The motion generated from the learned model can be scaled in space and time. Besides, a neural-network-based controller is designed for the robot to track the trajectories generated from the motion model. In this controller, a radial basis function neural network is used to compensate for the effect caused by the dynamic environments. The experiments have been performed using a Baxter robot and the results have confirmed the validity of the proposed methods.**

***Index Terms*—Dynamic movement primitives (DMPs), Gaussian mixture model (GMM), neural network (NN), robot learning.**

I. Introduction

RECENTLY, robots have been widely applied in various fields, especially in manufacturing. Adaptable robots are required due to the increasingly fast updates of the manufactured products. Hence, it is necessary to develop methods for enhancing robot learning. Robot learning from demonstration (LfD) is a valuable technique to simplify the strategy of robot learning [1], [2]. The human tutor shows the way to complete a task and then the robot learns, via motion modeling, to reproduce the skill. Therefore, it is essential to consider how to model motions effectively.

The dynamic system (DS) is a powerful tool for motion modeling [3]. Compared to the conventional methods, e.g., interpolation techniques, DS offers a flexible solution to model stable and extensible trajectories. In addition, the motion encoded with the DS is robust to perturbations. An approach based on DS was used to learn human motions [4], where the unknown mapping of the DS was approximated using a neural network (NN) called extreme learning machine [5]. The learned model showed adequate stability and generalization. However, this DS-based method required considerable demonstration data for training. In contrast, the dynamic movement primitive (DMP), which is based on a nonlinear DS [6], only requires one demonstration to model motion; here, the DMP models the movement trajectory as a spring-damper system integrated with an unknown function to be learned. The inherent property of the spring-damper system enhances the stability and robustness (to perturbations) of the generated motion.

DMPs have been often employed to solve robot learning problems because of their flexibility. In [7], DMPs were modified to model fast movement inherent in hitting motion. Another study used reinforcement learning to combine DMP sequences so that the robot could perform more complex tasks [8]. While both these studies employed multiple DMPs to compose a complete action, another study [9] used multiple DMPs to model style-adaptive trajectory, where the style of the generated motion could be changed by modulating the weight parameters that were coupled with the goals. As mentioned in [10], optimal demonstration is difficult to obtain and multiple demonstrations can encode the ideal trajectory implicitly. Therefore, we consider integrating multiple demonstrations into one DMP model in this paper.

Probabilistic approaches have shown good performance in motion encoding [11]–[13]. The inherent variability of the demonstrations can be extracted, and thus, more features of the demonstrations can be preserved. In [14], an LfD framework using a Gaussian mixture model (GMM) and a Bernoulli mixture model was used to extract the features from multiple demonstrations. A new motion was generated through Gaussian mixture regression (GMR). In contrast with the above-mentioned methods, GMM combined with GMR can provide additional motion information for robots when

Manuscript received August 17, 2017; revised December 17, 2017 and March 19, 2018; accepted June 22, 2018. Date of publication July 26, 2018; date of current version February 19, 2019. This work was supported in part by the National Nature Science Foundation under Grant 61473120 and Grant 61472325, in part by the Science and Technology Planning Project of Guangzhou under Grant 201607010006, in part by the State Key Laboratory of Robotics and System under Grant SKLRS-2017-KF-13, and in part by Fundamental Research Funds for the Central Universities under Grant 2017ZD057. *(Corresponding author: Chenguang Yang.)*

C. Yang and C. Chen are with the Key Laboratory of Autonomous Systems and Networked Control, College of Automation Science and Engineering, South China University of Technology, Guangzhou 510640, China (e-mail: cyang@ieee.org; aucchen@outlook.com).

W. He is with the School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: weihe@ieee.org).

R. Cui is with the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: rongxin.cui@gmail.com).

Z. Li is with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: zjli@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2852711

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see http://creativecommons.org/licenses/by/3.0/

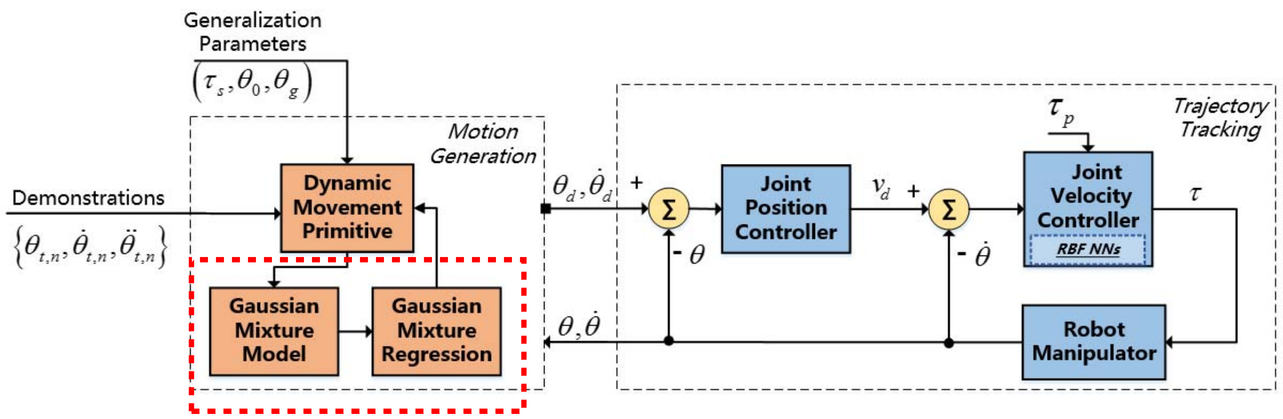

Generalization Parameters
$(\tau_s, \theta_0, \theta_g)$
Motion Generation
Demonstrations
$\{\theta_{t,n}, \dot{\theta}_{t,n}, \ddot{\theta}_{t,n}\}$
Dynamic Movement Primitive
Gaussian Mixture Model
Gaussian Mixture Regression
$\theta_d, \dot{\theta}_d$
$\theta, \dot{\theta}$
+
$-\theta$
Joint Position Controller
v_d
+
$-\dot{\theta}$
τ_p
Trajectory Tracking
Joint Velocity Controller
RBF NNs
τ
Robot Manipulator

Fig. 1. Block diagram of the proposed system.

learning from multiple demonstrations. In [3], a learning approach named stable estimator of dynamical systems (SEDS) was proposed for motion modeling, where an unknown function was modeled using GMR. DS-GMR is another method that combines the DS with the statistical learning approach [15]. Both methods exploit the robustness and generalization capability of the DS as well as the excellent learning performance of the probabilistic methods.

To take advantage of the performance of the DS and the probabilistic approach, we integrate DMP and GMM into our proposed system, where the nonlinear function of DMP is modeled with GMM and its estimate is retrieved through GMR. This modification enables the robot to extract more features of the motions from multiple demonstrations and to generate motions that synthesize these features. The original DMP was learned using the locally weighted regression (LWR) [16], and the locally weighted projection regression [17] was employed to optimize the bandwidth of each kernel of LWR. Despite the added complexity of the learning procedure, these methods enable the DMP to learn from only one demonstration. Reservoir computing [18] is another method used to approximate the nonlinear function, but its computing efficiency is less than that of GMR.

The imitation performance of robots also depends on the accuracy of the trajectory tracking controller that involves the robot dynamics. Generally, a model-based control performs better if the model is accurate enough [19]. However, an accurate dynamic model of a manipulator cannot be obtained in advance due to some uncertainties, e.g., unknown payload. The approximation-based controllers have been designed to overcome such uncertainties. They utilize function approximation tools to learn the nonlinear characteristics of the robot dynamics. NNs have been widely used in controller design because of their approximation ability [20]–[22]. In [23], the backpropagation NN (BPNN) was utilized to approximate the unknown nonlinear function in the model of the vibration suppression device, while in [24], the radial basis function NN (RBFNN) was utilized to approximate the unknown nonlinearity of the telerobot system. Compared to BPNN, the learning procedure of RBFNN is based on local approximation; thus, RBFNN can avoid getting stuck in the local optimum and has a faster convergence rate. Besides, the number of hidden layer units of RBFNN can be adaptively adjusted during the training phase, making NN more flexible and adaptive. Therefore, RBFNN is more appropriate for the design of real-time control.

In this paper, an NN-based controller is designed to guarantee the tracking performance of the manipulator in joint space, where RBFNN is employed to approximate the nonlinear functions of the robot dynamics. The stability of the controller is guaranteed by the Lyapunov stability theory. As shown in Fig. 1, the robot learning system consists of the motion generation component and the trajectory tracking component. The former utilizes the motion model based on DMP to learn and generalize motion skills; these, in turn, are represented as a set of trajectories in joint space. The latter employs the adaptive controller to track the trajectories generated from the former, and RBFNN is incorporated to compensate for the uncertain dynamics.

Here, we present a novel and complete robot learning framework that considers the performance of both motion generation and trajectory tracking. The SEDS presented in [3] is similar to our DMP-based model. However, the constraints that guarantee the stability of SEDS are derived by the Lyapunov theory that increases the complexity of the learning. In contrast to [3] and [25] which considered only motion modeling, our system is enhanced by an NN-based controller and the effect caused by the dynamic environments can be compensated by neural learning. This design enables the robot to perform the learned motions steadily and more robustly in the real world.

The remainder of this paper is organized as follows. Section II introduces the DMP and its relevant characteristics. The learning process of the motion model is introduced in Section III. In Section IV, the concept of RBFNN is introduced, and the controller using RBFNN is designed with the proof of stability. The experiments are presented in Section V. Section VI concludes this paper.

II. Basic Model of Discrete Movement

Motion skills can be classified into point-to-point and periodic movements in accordance with brain activation [1]. While using DS to model motions, these two types correspond to point attractor and limit cycle attractor, respectively.

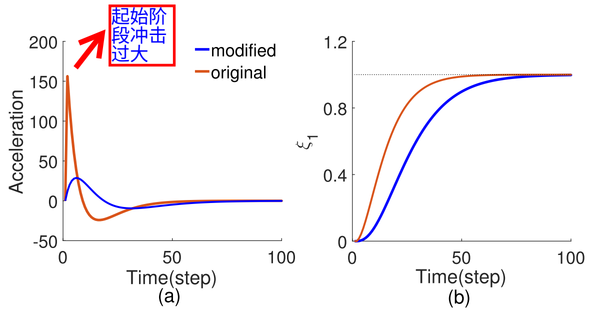


Fig. 2. (a) Accelerations of the original spring-damper part in (1) and the modified one in (4). (b) Evolutions of the systems (1) and (4) without an external forcing term ($f = 0$).

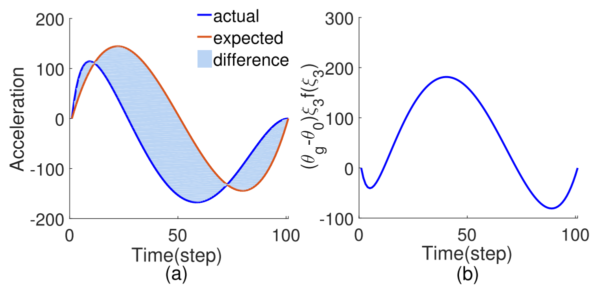


Fig. 3. (a) Actual acceleration of the spring-damper part and the expected acceleration to reproduce the demonstration (a sine curve). (b) Difference between two accelerations in (a), which is compensated by the nonlinear function $f(\xi_3)$.

In this paper, we focus on point-to-point movements and use the discrete DMP as the basic motion model, which is composed of a spring-damper system and a nonlinear function.

DMP can be employed to model motions in joint space or Cartesian space. The motions in both spaces are regarded as a set of 1-D trajectories, and each trajectory is represented as one DMP model. For concision, we only discuss the modeling problem of the motions in joint space.

The DMP model is defined as follows [26]:

$$\tau_s\dot{\xi}_1 = \xi_2$$
$$\tau_s\dot{\xi}_2 = l_1(\theta_g - \xi_1) - l_2\xi_2 + (\theta_g - \theta_0)\xi_3 f(\xi_3) \quad (1)$$
$$\tau_s\dot{\xi}_3 = -\alpha_1\xi_3 \quad (2)$$

where $\xi_1 \in R$ represents the joint position and $\xi_2/\tau_s \in R$ and $\dot{\xi}_2/\tau_s \in R$ denote the joint velocity and the joint acceleration, respectively. $\xi_3 > 0$ is regarded as a phase variable that is exponentially decaying. l_1, l_2, and α_1 are the positive constants. θ_0 is the start position, θ_g is the goal, and $(\theta_g - \theta_0)$ serves as the spatial scaling term. $\tau_s > 0$ is the time constant. The nonlinear term $f : R \to R$ is assumed to be a continuous bounded function in terms of ξ_3.

The system (1) is a spring-damper system perturbed by a nonlinear term. Generally, we choose $l_1 = l_2^2/4$ to make the former critically damped. The initial state of system (2) can be chosen as $\xi_0 = 1$. The stability of the whole model is clear; the nonlinear term will converge to zero since the state ξ_3 will converge to zero and the nonlinear function $f(\xi_3)$ is bounded. Then, the system (1) becomes a stable spring-damper system, whose state converges to the goal θ_g.

The large initial acceleration of the original spring-damper part [Fig. 2(a)] is not desirable for robots in practice. Moreover, the large variation in the acceleration will lead to a complex external forcing term, which is adverse to the learning of the model. Therefore, we replace the goal θ_g in the spring-damper part with the state of another exponential decay system [27]

$$\tau_s\dot{\xi}_4 = -\alpha_2(\xi_4 - \theta_g) \quad (3)$$

where α_2 is a positive constant and the initial value of ξ_4 is set as θ_0. Consequently, the system (1) is rewritten as follows [27]:

$$\tau_s\dot{\xi}_1 = \xi_2$$
$$\tau_s\dot{\xi}_2 = l_1(\xi_4 - \xi_1) - l_2\xi_2 + (\theta_g - \theta_0)\xi_3 f(\xi_3). \quad (4)$$

As shown in Fig. 2(a), the acceleration of the modified system becomes moderate. The evolution of the system (4) with $f = 0$ is shown in Fig. 2(b). Since the state ξ_4 will converge to θ_g, the modified model is still stable toward the goal.

The DMP model is chosen as the basic motion model because of its concise formulation and excellent generalizability. Since the DMP model is always stable toward θ_g, the motion modeled can be scaled spatially by modulating θ_g and θ_0 without destabilizing the model. The spatial scaling term $(\theta_g - \theta_0)$ ensures that the profile of the motion is maintained and its duration can be changed by modulating τ_s.

III. Learning of the Motion Model

A. Problem Description

DMP assumes that the motion is generated from the nonlinear DS (4), whose uncertain part is the nonlinear function $f(\xi_3)$. Therefore, the learning problem of DMP is how to approximate this function, which compensates for the difference between the actual acceleration of the spring-damper part and the expected acceleration to reproduce the demonstrated movement (Fig. 3). Assuming that a 1-D demonstration trajectory is captured and encoded as a time series $\{(\theta_t, \dot{\theta}_t, \ddot{\theta}_t) \mid t = 1, 2, \ldots, T\}$, where θ_t, $\dot{\theta}_t$, and $\ddot{\theta}_t$ describe the position, velocity, and acceleration of the trajectory at time step t, respectively, and T is the duration of the demonstration, the expected value of f at time step t is computed using the following equation [26]:

$$\boxed{f_t = \frac{\tau_s^2\ddot{\theta}_t - [l_1(\xi_{4,t} - \theta_t) - l_2\tau_s\dot{\theta}_t]}{(\theta_T - \theta_1)\xi_{3,t}}} \quad (5)$$

where $\xi_{3,t}$ and $\xi_{4,t}$ are the values of ξ_3 and ξ_4 at time step t, respectively. Equation (5) is an inverse process of the reproduction. Since $f(\xi_3)$ is a function of ξ_3, we can exploit the data set $\{(\xi_{3,t}, f_t) \mid t = 1, 2, \ldots, T\}$ to approximate the function.

In the original DMP model, the function $f(\xi_3)$ is defined as [26]

$$f(\xi_3) = \sum_{i=1}^{N_s} \gamma_i\phi_i(\xi_3) \quad (6)$$

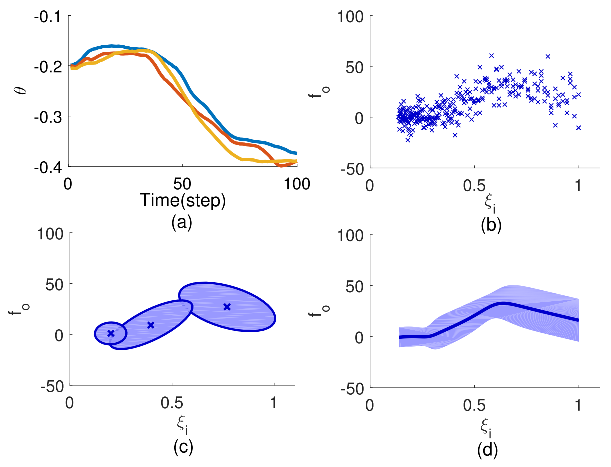


Fig. 4. (a) Demonstrations. (b) Data set $\{\xi_i, f_o\}$ calculated from (a). (c) Data set is encoded with the GMM. (d) Estimate of the function $f(\xi_3)$ is retrieved using the GMR.

where $\gamma_i \in R$ is the weight of $\phi_i(s)$ and $\phi_i(s)$ is the normalized Gaussian basis function, defined as

$$\phi_i(\xi_3) = \frac{\exp(-h_i(\xi_3 - c_i)^2)}{\sum_{j=1}^{N_s} \exp(-h_j(\xi_3 - c_j)^2)} \tag{7}$$

where $h_i > 0$ is the width and $c_i > 0$ is the center of the ith Gaussian functions; N_s is the number of the Gaussian functions. With this formulation, the LWR can be employed to learn the model. Nevertheless, the solution is only applicable to a single demonstration, and multiple demonstrations will be encoded as multiple independent DMP models. Thus, the features of the demonstrations of multiple specific tasks cannot be integrated into one DMP model. To solve this problem, GMM is introduced to model the intermediate data, and then GMR is employed to estimate the nonlinear function.

B. Learning From Multiple Demonstrations

Given N_d demonstrations $\{(\theta_{t,n}, \dot{\theta}_{t,n}, \ddot{\theta}_{t,n}) \mid t = 1, 2, \ldots, T; n = 1, 2, \ldots, N_d\}$ of a specific task, we can obtain the data set $\{(\xi_{3,t}, f_{t,n}) \mid t = 1, 2, \ldots, T; n = 1, 2, \ldots, N_d\}$ using (5). Then, we need to take advantage of the whole data set to estimate the function $f(\xi_3)$ of one DMP model. The learning procedures can be broken down into two steps, as shown in Fig. 4. For concision, we use $\{\xi_i, f_o\}$ to denote the data set.

Step 1: Use GMM to encode the data set $\{\xi_i, f_o\}$. This model assumes that the data set is generated from multiple Gaussian distributions. The joint probability density of GMM is defined as follows [28]:

$$p(\xi_i, f_o) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\xi_i, f_o; \mu_k, \Sigma_k) \tag{8}$$

where

$$\sum_{k=1}^{K} \alpha_k = 1 \tag{9}$$

and

$$\mu_k = \begin{bmatrix} \mu_{\xi_i,k} \\ \mu_{f_o,k} \end{bmatrix}, \quad \Sigma_k = \begin{bmatrix} \Sigma_{\xi_i,k} & \Sigma_{\xi_i f_o,k} \\ \Sigma_{f_o \xi_i,k} & \Sigma_{f_o,k} \end{bmatrix} \tag{10}$$

and $\mathcal{N}$ is the Gaussian probability distribution defined as follows:

$$\mathcal{N}(\xi_i, f_o; \mu_k, \Sigma_k) = \frac{\exp[-0.5([\xi_i, f_o]^T - \mu_k)^T \Sigma_k^{-1}([\xi_i, f_o]^T - \mu_k)]}{2\pi\sqrt{|\Sigma_k|}}. \tag{11}$$

Here, K is the number of Gaussian distributions, $\alpha_k \geq 0$ is the weight, and $\mu_k \in R^{2\times1}$ and $\Sigma_k \in R^{2\times2}$ denote the mean and the covariance matrix of the kth Gaussian component.

α_k, μ_k, and Σ_k are the parameters to be learned. We can estimate their values and learn GMM from the given data set using the following procedures.

1) Parameter Initialization: The expectation–maximization (EM) algorithm used for parameter estimation is sensitive to the initial values. Hence, the k-means algorithm [29] is utilized to initialize the unknown parameters α_k, μ_k, and Σ_k. This algorithm divides the data set into multiple sets and aims to find a partition $D = \{D_1, D_2, \ldots, D_K\}$ to minimize the sum of the squared deviation of each set [29]

$$\hat{D} = \arg\min_D \sum_{k=1}^{K} \sum_{x \in D_k} \|x - m_k\|^2 \tag{12}$$

where $x = [x_1, x_2]^T \in R^{2\times1}$ is the data vector corresponding to $[\xi_{3,t}, f_{t,n}]^T$, $m_k \in R^{2\times1}$ is the mean of the data distributed in D_k, and $\bigcup_{k=1}^{K} D_k = \{\xi_i, f_o\}$. The algorithm repeats the assignment and update steps until that partition no longer changes. Then, the initial values of the unknown parameters are assigned as [14]

$$\alpha_k = \frac{|D_k|}{\sum_{i=1}^{K} |D_i|} \tag{13}$$

$$\mu_k = m_k, \quad \Sigma_k = \begin{bmatrix} \Sigma_{x_1} & \Sigma_{x_1 x_2} \\ \Sigma_{x_2 x_1} & \Sigma_{x_2} \end{bmatrix}_{x \in D_k}. \tag{14}$$

2) Parameter Estimation: The EM algorithm is an appropriate method for estimating the parameters of GMM. This algorithm aims to find parameters $\pi_k = (\alpha_k, \mu_k, \Sigma_k)$ so as to maximize the log-likelihood function [30]

$$\hat{\pi}_k = \arg\max_{\pi_k} \log(p(\xi_i, f_o | \pi_k)). \tag{15}$$

3) Model Selection: The number of the Gaussian components will affect the fitting performance of GMM. GMM can represent the data set better if more components are selected, but it may lead to overfitting and too many parameters. A compromise is reached using the Bayesian information criterion (BIC) to determine the number of components. The BIC score is defined as follows [14]:

$$S_{BIC}(K) = -2\log(p(\xi_i, f_o | \pi_k)) + (6K - 1)\log(N_d) \tag{16}$$

where N_d is the size of the data set and K represents the number of components. The number of components is finally selected as $K_{ms} = \arg\min S_{BIC}(K)$, and the upper bound of K_{ms} is determined by the designer.

Step 2: Use GMR to estimate the function $f(\xi_3)$. According to [28], the probability density (8) can be rewritten as

$$p(\xi_i, f_o) = \sum_{k=1}^{K} \alpha_k \mathcal{N}\big(f_o; \hat{\eta}_k, \hat{\sigma}_k^2\big)\mathcal{N}(\xi_i; \mu_{\xi_i,k}, \Sigma_{\xi_i,k}) \tag{17}$$

where

$$\hat{\eta}_k(\xi_i) = \mu_{f_o,k} + \Sigma_{f_o\xi_i,k}(\Sigma_{\xi_i,k})^{-1}(\xi_i - \mu_{\xi_i,k}) \tag{18}$$

and

$$\hat{\sigma}_k^2 = \Sigma_{f_o,k} - \Sigma_{f_o\xi_i,k}(\Sigma_{\xi_i,k})^{-1}\Sigma_{\xi_i f_o,k}. \tag{19}$$

The marginal density of ξ_i is calculated as follows [28]:

$$\begin{aligned} p(\xi_i) &= \int p(\xi_i, f_o)df_o \\ &= \sum_{k=1}^{K} \alpha_k \mathcal{N}(\xi_i; \mu_{\xi_i,k}, \Sigma_{\xi_i,k}). \end{aligned} \tag{20}$$

According to (17) and (20), the conditional probability density of f_o given ξ_i is

$$p(f_o|\xi_i) = \sum_{k=1}^{K} \beta_k(\xi_i)\mathcal{N}(f_o; \hat{\eta}_k, \hat{\sigma}_k^2) \tag{21}$$

where

$$\beta_k(\xi_i) = \frac{\alpha_k \mathcal{N}(\xi_i; \mu_{\xi_i,k}, \Sigma_{\xi_i,k})}{\sum_{k=1}^{K} \alpha_k \mathcal{N}(\xi_i; \mu_{\xi_i,k}, \Sigma_{\xi_i,k})}. \tag{22}$$

Then, we can obtain the GMR function as follows [28]:

$$R(\xi_i) = E(f_o|\xi_i) = \sum_{k=1}^{K} \beta_k(\xi_i)\hat{\eta}_k(\xi_i). \tag{23}$$

In addition, the estimate of function $f(\xi_3)$ is

$$\hat{f}(\xi_3) = \sum_{k=1}^{K} \beta_k(\xi_3)\hat{\eta}_k(\xi_3). \tag{24}$$

In comparison to (6), this estimate is multiplied by an additional term $\hat{\eta}_k(\xi_3)$ in form; thus, this method enables the motion model to extract more features from multiple demonstrations.

IV. Adaptive Neural Control of the Robot Manipulator

In practice, robot manipulators have to interact with various payloads. To account for the influence of the unknown payload on the manipulator dynamics, an NN-based controller is designed to track the reference trajectory generated from the motion model in joint space.

A. Dynamics Description

The dynamics of a robot manipulator with n-link is described as follows:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G_0(\theta) + \tau_p = \tau \tag{25}$$

where $\theta \in R^n$, $\dot{\theta} \in R^n$, and $\ddot{\theta} \in R^n$ are the joint position, joint velocity, and joint acceleration, respectively. τ is the control torque and τ_p is the torque caused by the payload. $M(\theta) \in R^{n\times n}$ denotes the inertia matrix, which is symmetric positive definite. $C(\theta, \dot{\theta}) \in R^{n\times n}$ represents the Coriolis and centripetal torque matrix and $G_0(\theta) \in R^n$ is the gravity vector. According to [31], the matrices $M(\theta)$ and $C(\theta, \dot{\theta})$ satisfy

$$s^T(\dot{M} - 2C)s = 0, \quad \forall s \in R^n. \tag{26}$$

B. RBFNN

RBFNN is an effective tool to approximate any continuous function $g : R^m \to R$ as follows [32]:

$$g(\vartheta) = W^T S(\vartheta) + \varepsilon(\vartheta), \quad \forall \vartheta \in \Omega_\vartheta \tag{27}$$

where $\vartheta \in \Omega_\vartheta \subset R^m$ denotes the input vector, $W = [\omega_1, \omega_2, \ldots, \omega_N]^T \in R^N$ is the ideal NN weight vector, and N is the number of NN nodes. The approximation error $\varepsilon(\vartheta)$ is bounded. $S(\vartheta) = [s_1(\vartheta), s_2(\vartheta), \ldots, s_N(\vartheta)]^T$ is a nonlinear vector function, where $s_i(\vartheta)$ is defined as the Gaussian function

$$s_i(\vartheta) = \exp\left[-\frac{(\vartheta - \kappa_i)^T(\vartheta - \kappa_i)}{\chi_i^2}\right], \quad i = 1, 2, \ldots, N \tag{28}$$

where $\kappa_i = [\kappa_{i1}, \kappa_{i2}, \ldots, \kappa_{im}]^T \in R^m$ represents the center of the Gaussian function and χ_i^2 is the variance. The ideal weight vector W is defined as follows:

$$W = \arg\min_{\hat{W}\in R^N}\{\sup_{\vartheta\in\Omega_\vartheta} |g(\vartheta) - \hat{W}^T S(\vartheta)|\} \tag{29}$$

which minimizes the approximation error for all $\vartheta \in \Omega_\vartheta$.

C. Controller Design

The controller design includes the design of the joint position controller and the joint velocity controller, as shown in Fig. 1. RBFNN is used in the latter to approximate the uncertain dynamics.

1) Joint Position Controller: The joint position tracking error is defined as $e_p = [e_{p1}, e_{p2}, \ldots, e_{pn}]^T = \theta - \theta_d$, where $\theta_d = [\theta_{d1}, \theta_{d2}, \ldots, \theta_{dn}]^T \in R^n$ is the reference trajectory, which is smooth and bounded. The error transformation function [33] is introduced as follows:

$$e_{pi}(t) = \delta(t)H_i\left(L_i\left(\frac{e_{pi}(t)}{\delta(t)}\right)\right), \quad i = 1, 2, \ldots, n \tag{30}$$

where $\delta(t) = (\delta_0 - \delta_\infty)e^{-at} + \delta_\infty$ represents the tracking performance requirement, and $H_i(z)$ is defined as

$$H_i(z) = \begin{cases} \dfrac{e^z - \sigma}{1 + e^z}, & e_{pi}(0) \ge 0 \\ \dfrac{\sigma e^z - 1}{1 + e^z}, & e_{pi}(0) < 0 \end{cases} \tag{31}$$

and $L_i(z)$ is the inverse function of $H_i(z)$

$$L_i(z) = \begin{cases} \ln \dfrac{z+\sigma}{1-z}, & e_{pi}(0) \geq 0 \\ \ln \dfrac{z+1}{\sigma - z}, & e_{pi}(0) < 0. \end{cases} \tag{32}$$

The parameters δ_0, $\delta_\infty < \delta_0$, a, and σ are the positive constants, which are used to adjust the control performance. The joint position controller is used to generate the desired joint velocity, which is designed as [24]

$$\upsilon_{di} = -k_1\delta(t)\phi_i(t) + \dot{\theta}_{di}(t) + \frac{\dot{\delta}(t)}{\delta(t)}e_{pi}(t) \tag{33}$$

where

$$\phi_i(t) = L_i\left(\frac{e_{pi}(t)}{\delta(t)}\right). \tag{34}$$

According to [24], if $\phi_i(t)$ is bounded, then the following is obtained:

$$\begin{cases} -\sigma\delta(t) < e_{pi}(t) < \delta(t), & e_{pi}(0) > 0 \\ -\delta(t) < e_{pi}(t) < \sigma\delta(t), & e_{pi}(0) < 0. \end{cases} \tag{35}$$

Therefore, the function $\delta(t)$ determines the bound of error $e_{pi}(t)$ and the transient performance of the controller.

2) *Joint Velocity Controller:* The joint velocity controller aims to generate the control torque so as to track the desired joint velocity $\upsilon_d = [\upsilon_{d1}, \upsilon_{d2}, \ldots, \upsilon_{dn}]^T$. Let us define the joint velocity error as $e_\upsilon = \dot{\theta} - \upsilon_d$ and $G(\theta) = G_0(\theta) + \tau_p$. Then, the control torque is designed as follows [24]:

$$\tau = -k_2 e_\upsilon - \frac{\dot{Q}(\phi(t))\phi(t)}{\delta(t)} + \hat{M}\dot{\upsilon}_d + \hat{C}\upsilon_d + \hat{G} + \hat{r} \tag{36}$$

where

$$\begin{aligned} \dot{Q}(\phi(t)) &= \mathrm{diag}(\dot{L}_1(H_1(\phi_1(t))), \ldots, \dot{L}_n(H_n(\phi_n(t)))) \\ \phi(t) &= [\phi_1(t), \phi_2(t), \ldots, \phi_n(t)]^T. \end{aligned} \tag{37}$$

The matrices $\hat{M}(\theta)$, $\hat{C}(\theta,\dot{\theta})$, $\hat{G}(\theta)$, and $\hat{r}(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d)$ are the estimates of $M(\theta)$, $C(\theta,\dot{\theta})$, $G(\theta)$, and $r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d)$, respectively, where $r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d)$ is defined later in (40).

Substituting (36) into (25), we can obtain the closed-loop dynamics equation

$$\begin{aligned} &M\dot{e}_\upsilon + Ce_\upsilon + k_2 e_\upsilon + \frac{\dot{Q}(\phi(t))\phi(t)}{\delta(t)} - \hat{r} \\ &\quad = -(M - \hat{M})\dot{\upsilon}_d - (C - \hat{C})\upsilon_d - (G - \hat{G}). \end{aligned} \tag{38}$$

Then, RBFNN is utilized to approximate $M(\theta)$, $C(\theta,\dot{\theta})$, $G(\theta)$, and $r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d)$

$$\begin{aligned} M(\theta) &= W_M^T S_M(\theta) + \varepsilon_M \\ C(\theta,\dot{\theta}) &= W_C^T S_C(\theta,\dot{\theta}) + \varepsilon_C \\ G(\theta) &= W_G^T S_G(\theta) + \varepsilon_G \\ r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d) &= W_r^T S_r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d) + \varepsilon_r \end{aligned} \tag{39}$$

where $W_M \in R^{nN\times n}$, $W_C \in R^{2nN\times n}$, $W_G \in R^{nN\times n}$, and $W_r \in R^{4nN\times n}$ are the ideal NN weight matrices. $S_M(\theta) \in R^{nN\times n}$, $S_C(\theta,\dot{\theta}) \in R^{2nN\times n}$, $S_G(\theta) \in R^{nN\times n}$, and $S_r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d) \in R^{4nN\times n}$ are the RBF matrices. ε_M, ε_C, ε_G, and ε_r are the approximation errors. The function $r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d)$ is defined as

$$r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d) = \varepsilon_M\dot{\upsilon}_d + \varepsilon_C\upsilon_d + \varepsilon_G. \tag{40}$$

The estimates of M, C, G, and r are written as follows:

$$\begin{aligned} \hat{M}(\theta) &= \hat{W}_M^T S_M(\theta) \\ \hat{C}(\theta,\dot{\theta}) &= \hat{W}_C^T S_C(\theta,\dot{\theta}) \\ \hat{G}(\theta) &= \hat{W}_G^T S_G(\theta) \\ \hat{r}(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d) &= \hat{W}_r^T S_r(\theta,\dot{\theta},\upsilon_d,\dot{\upsilon}_d). \end{aligned} \tag{41}$$

Substituting (41) into (38) and defining $\tilde{W}_{(\cdot)} = W_{(\cdot)} - \hat{W}_{(\cdot)}$, we have

$$\begin{aligned} &M\dot{e}_\upsilon + Ce_\upsilon + k_2 e_\upsilon + \frac{\dot{Q}(\phi(t))\phi(t)}{\delta(t)} \\ &\quad = -\tilde{W}_M^T S_M\dot{\upsilon}_d - \tilde{W}_C^T S_C\upsilon_d - \tilde{W}_G^T S_G - \tilde{W}_r^T S_r - \varepsilon_r. \end{aligned} \tag{42}$$

D. Stability Analysis

We follow the stability analysis in [24] to prove the stability of the designed controller. Consider a Lyapunov function as follows:

$$V = V_1 + V_2 \tag{43}$$

with

$$V_1 = \frac{1}{2}\phi^T(t)\phi(t) \tag{44}$$

and

$$\begin{aligned} V_2 = \frac{1}{2}e_\upsilon^T M e_\upsilon &+ \frac{1}{2}\mathrm{tr}\left(\tilde{W}_M^T\Gamma_M^{-1}\tilde{W}_M + \tilde{W}_C^T\Gamma_C^{-1}\tilde{W}_C\right) \\ &+ \frac{1}{2}\mathrm{tr}\left(\tilde{W}_G^T\Gamma_G^{-1}\tilde{W}_G + \tilde{W}_r^T\Gamma_r^{-1}\tilde{W}_r\right) \end{aligned} \tag{45}$$

where Γ_M^{-1}, Γ_C^{-1}, Γ_G^{-1}, and Γ_r^{-1} are the positive definite matrices.

Taking the derivatives of V_1 and V_2, we have

$$\dot{V}_1 = \frac{\phi^T(t)\dot{Q}(\phi(t))e_\upsilon(t)}{\delta(t)} - k_1\phi^T(t)\dot{Q}(\phi(t))\phi(t) \tag{46}$$

and

$$\begin{aligned} \dot{V}_2 = &-e_\upsilon^T k_2 e_\upsilon - e_\upsilon^T\varepsilon_r - \frac{\phi^T(t)\dot{Q}(\phi(t))e_\upsilon(t)}{\delta(t)} \\ &- \mathrm{tr}\left[\tilde{W}_M^T\left(S_M\dot{\upsilon}_d e_\upsilon^T + \Gamma_M^{-1}\dot{\hat{W}}_M\right)\right] \\ &- \mathrm{tr}\left[\tilde{W}_C^T\left(S_C\upsilon_d e_\upsilon^T + \Gamma_C^{-1}\dot{\hat{W}}_C\right)\right] \\ &- \mathrm{tr}\left[\tilde{W}_G^T\left(S_G e_\upsilon^T + \Gamma_G^{-1}\dot{\hat{W}}_G\right)\right] \\ &- \mathrm{tr}\left[\tilde{W}_r^T\left(S_r e_\upsilon^T + \Gamma_r^{-1}\dot{\hat{W}}_r\right)\right]. \end{aligned} \tag{47}$$

Let us design the update law of the NN weights as follows:

$$\begin{aligned} \dot{\hat{W}}_M &= -\Gamma_M\left(S_M\dot{\upsilon}_d e_\upsilon^T + \rho_M\hat{W}_M\right) \\ \dot{\hat{W}}_C &= -\Gamma_C\left(S_C\upsilon_d e_\upsilon^T + \rho_C\hat{W}_C\right) \\ \dot{\hat{W}}_G &= -\Gamma_G\left(S_G e_\upsilon^T + \rho_G\hat{W}_G\right) \\ \dot{\hat{W}}_r &= -\Gamma_r\left(S_r e_\upsilon^T + \rho_r\hat{W}_r\right) \end{aligned} \tag{48}$$

where ρ_M, ρ_C, ρ_G, and ρ_r are the positive constants. Then, the derivative of V is written as follows:

$$\begin{aligned}\dot{V} &= -e_v^T k_2 e_v - e_v^T \varepsilon_r - k_1 \phi^T(t)\dot{Q}(\phi(t))\phi(t) \\ &\quad + \mathrm{tr}\left[\rho_M \tilde{W}_M^T \hat{W}_M\right] + \mathrm{tr}\left[\rho_C \tilde{W}_C^T \hat{W}_C\right] \\ &\quad + \mathrm{tr}\left[\rho_G \tilde{W}_G^T \hat{W}_G\right] + \mathrm{tr}\left[\rho_r \tilde{W}_r^T \hat{W}_r\right]. \end{aligned} \tag{49}$$

We can obtain the following inequality according to the definition of $\dot{Q}(\phi(t))$:

$$\phi^T(t)\dot{Q}(\phi(t))\phi(t) \geq \frac{2}{(1+\sigma)}\|\phi(t)\|^2. \tag{50}$$

Using Young's inequality, we have

$$\begin{aligned}\dot{V} \leq& -\frac{2k_1}{(1+\sigma)}\|\phi(t)\|^2 - \left(k_2 - \frac{1}{2}\right)\|e_v\|^2 + \varpi \\ &- \frac{\rho_M}{2}\|\tilde{W}_M\|_F^2 - \frac{\rho_C}{2}\|\tilde{W}_C\|_F^2 \\ &- \frac{\rho_G}{2}\|\tilde{W}_G\|_F^2 - \frac{\rho_r}{2}\|\tilde{W}_r\|_F^2 \end{aligned} \tag{51}$$

with

$$\begin{aligned}\varpi =& \frac{\rho_M}{2}\|W_M\|_F^2 + \frac{\rho_C}{2}\|W_C\|_F^2 + \frac{\rho_G}{2}\|W_G\|_F^2 \\ &+ \frac{\rho_r}{2}\|W_r\|_F^2 + \frac{1}{2}\kappa_r^2 \end{aligned} \tag{52}$$

where κ_r is the upper bound of $\|\varepsilon_r\|$ over $\mathbf{\Omega}$.

We have $\dot{V} \leq 0$ if $\tilde{W}_M$, $\tilde{W}_C$, $\tilde{W}_G$, $\tilde{W}_r$, $\phi(t)$, and e_v satisfy the inequality as follows:

$$\begin{aligned}\varrho =& \frac{2k_1}{(1+\sigma)}\|\phi(t)\|^2 + \left(k_2 - \frac{1}{2}\right)\|e_v\|^2 + \frac{\rho_M}{2}\|\tilde{W}_M\|_F^2 \\ &+ \frac{\rho_C}{2}\|\tilde{W}_C\|_F^2 + \frac{\rho_G}{2}\|\tilde{W}_G\|_F^2 + \frac{\rho_r}{2}\|\tilde{W}_r\|_F^2 \geq \varpi. \end{aligned} \tag{53}$$

According to the LaSalles theorem, all closed-loop signals of the DS composed of (25), (36), and (48) are semi-global uniformly bounded if the input signals θ_d and $\dot{\theta}_d$ are bounded. Besides, $\phi(t)$ and e_v will converge to an invariant set $\mathbf{\Omega}_i \subseteq \mathbf{\Omega}$ [24]

$$\begin{aligned}\mathbf{\Omega}_i =& \{(\|\phi(t)\|, \|e_v\|, \|W_M\|, \|W_C\|, \\ &\|W_G\|, \|W_r\|)|\varrho/\varpi \leq 1\}. \end{aligned} \tag{54}$$

Since the signal $\phi(t)$ is bounded, the transient performance and the stability of the controller are guaranteed.

V. Experiments

The proposed system is tested by two groups of experiments: the test of the NN-based controller and the test of the DMP-based motion model.

A. Test of the NN-Based Controller

In this group of experiments, the performance of NN learning is tested, which compensates for the uncertain manipulator dynamics caused by the payload. The experiments are performed on the Baxter robot that has two seven degrees-of-freedom arms. The joints from the shoulder to end-effector are named as $s0$, $s1$, $e0$, $e1$, $w0$, $w1$, and $w2$ in this paper. The experiment setup is shown in Fig. 5. The payload is attached using the left gripper of the robot, which

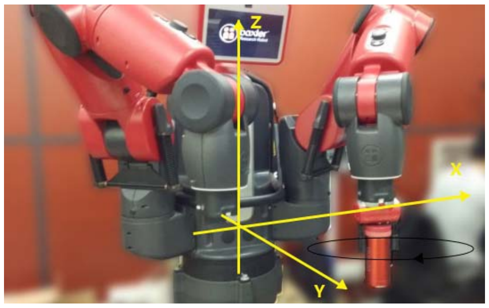
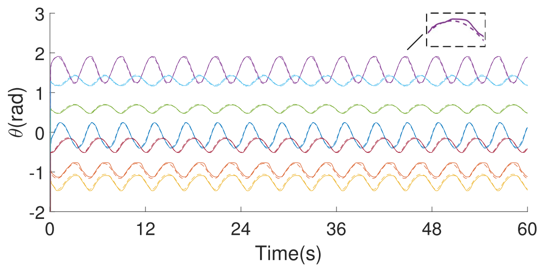
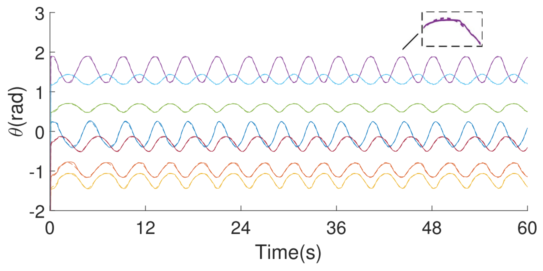


Fig. 5. Experimental setup for testing the NN-based controller.



(a)



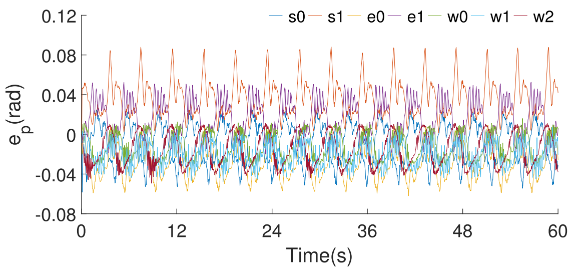
(b)

Fig. 6. Reference joint angles (dashed lines) and actual joint angles (solid lines) when NN learning is (a) disabled and (b) enabled. The lines of different colors represent different joints.

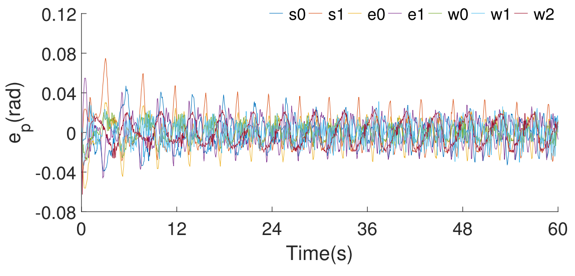
weighs 0.94 kg. The robot is required to track a circular trajectory defined as $[X, Y, Z] = [0.65 + 0.1\sin(2\pi t/4), 0.2 + 0.1\cos(2\pi t/4), 0.2]$(m) with the orientation fixed. The corresponding trajectories in joint space are obtained through the inverse kinematics, which are taken as the inputs of the proposed controller.

We select three nodes for each input dimension of NN, and the centers of the nodes are distributed evenly within the limits of the joint position and joint velocity. There are $N = 2187$ NN nodes selected for $\hat{M}(\theta)$ and $\hat{G}(\theta)$, $2N$ nodes for $\hat{C}(\theta, \dot{\theta})$, and $4N$ nodes for $\hat{r}(\theta, \dot{\theta}, \nu_d, \dot{\nu}_d)$. In addition, the NN weight matrices are initialized as $\hat{W}_M = \mathbf{0} \in R^{nN\times n}$, $\hat{W}_C = \mathbf{0} \in R^{2nN\times n}$, $\hat{W}_G = \mathbf{0} \in R^{nN\times n}$, and $\hat{W}_r = \mathbf{0} \in R^{4nN\times n}$ with $n = 7$. The parameters of the error transformation function are set as $\delta_0 = 0.2$, $\delta_\infty = 0.04$, $a = 1$, and $\sigma = 1$.

The manipulator is controlled by the controller without NN learning in the first experiment, and the actual joint angles are recorded. Then, the controller with the proposed NN learning is employed to control the manipulator in the second experiment. The reference and actual joint angles in both experiments are shown in Fig. 6, and the corresponding

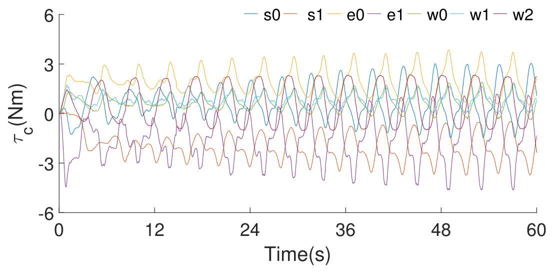


(a)

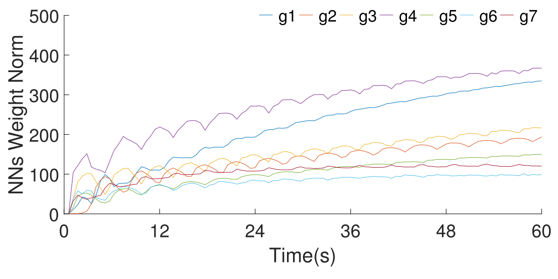


(b)

Fig. 7. Tracking errors when NN learning is (a) disabled and (b) enabled.



(a)



(b)

Fig. 8. (a) Compensation torque. (b) Norm of each column of the weight matrix $\hat{W}_G$.

tracking errors are shown in Fig. 7. The tracking errors are relatively high when NN learning is disabled, which is caused by the payload that the gripper holds, while in the second experiment, each joint of the manipulator tracks the reference trajectory very well and all tracking errors reduce into the interval $[-0.04, 0.04]$(rad) with the compensation torque increasing, which is shown in Fig. 8(a). The gravity term of the manipulator dynamics is the main part that the payload affects, and hence, we particularly show the norm of each column of $\hat{W}_G$ in Fig. 8(b). We can see that the norm of each column vector of the weight matrix $\hat{W}_G$ rises incrementally because the torque generated by NN still cannot compensate for the effect of the unknown dynamics. However, the rising speeds of all norms decrease with the increment in torque compensation.

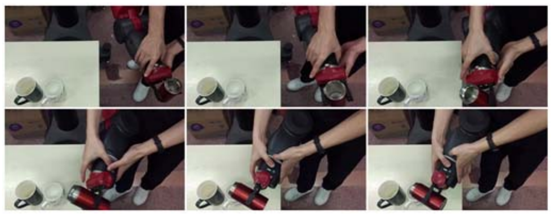
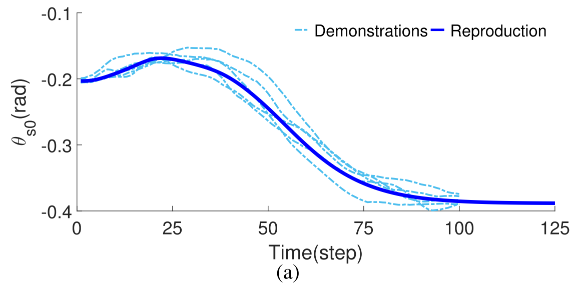
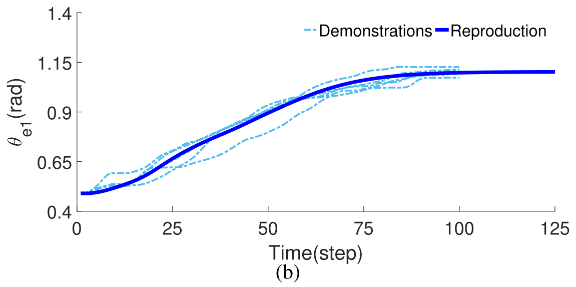


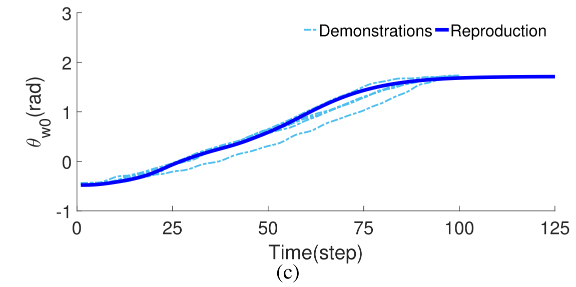
Fig. 9. Demonstration process of a pouring task.



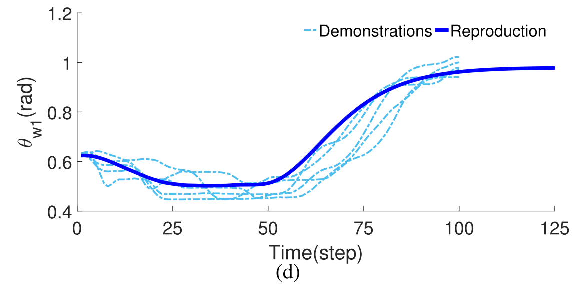
(a)



(b)



(c)



(d)

Fig. 10. Learning results using the DMP-based motion model in a pouring task of (a) joint $s0$, (b) joint $e1$, (c) joint $w0$, and (d) joint $w1$.

B. Test of the DMP-Based Motion Model

The second group of experiments aims to validate the DMP-based motion model. The ability of generalization

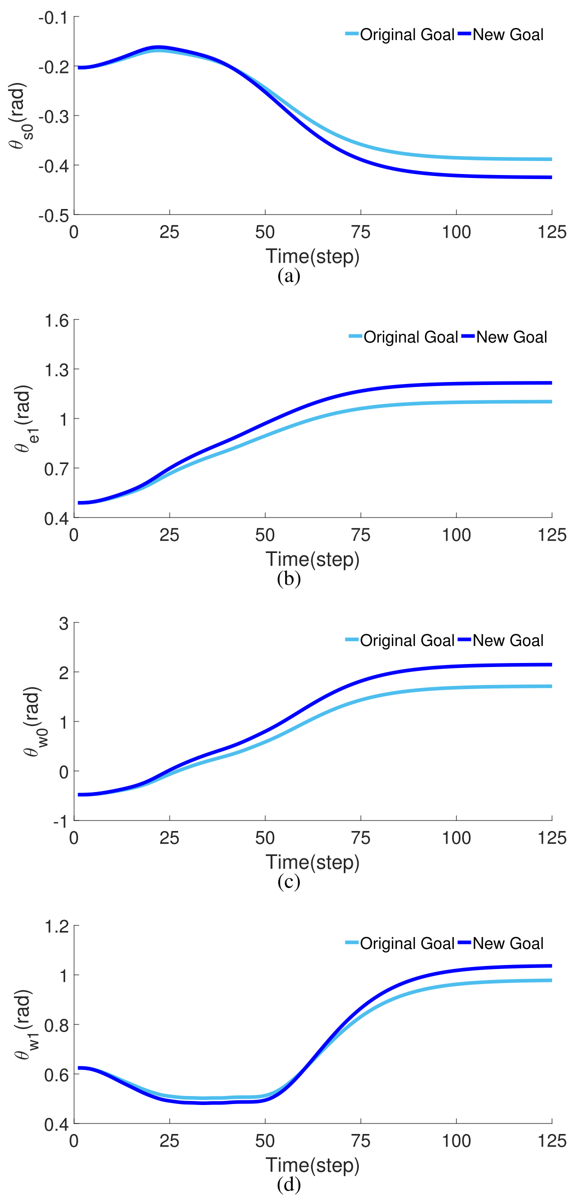


Fig. 11. Generalization results using the DMP-based motion model in a pouring task of (a) joint $s0$, (b) joint $e1$, (c) joint $w0$, and (d) joint $w1$.

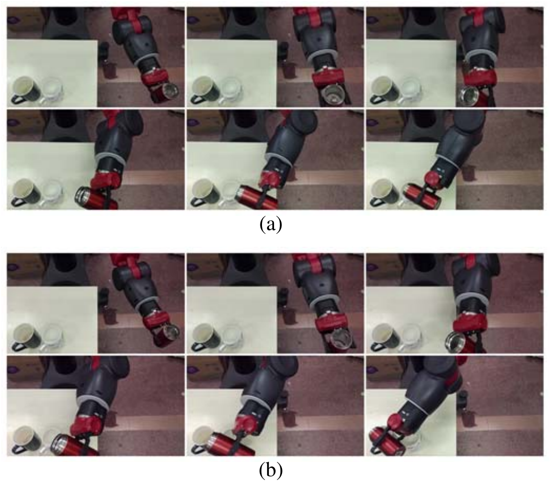


Fig. 12. (a) Robot performs the pouring task with the regenerated motion. (b) Robot pours water into the other cup with the generalized motion.

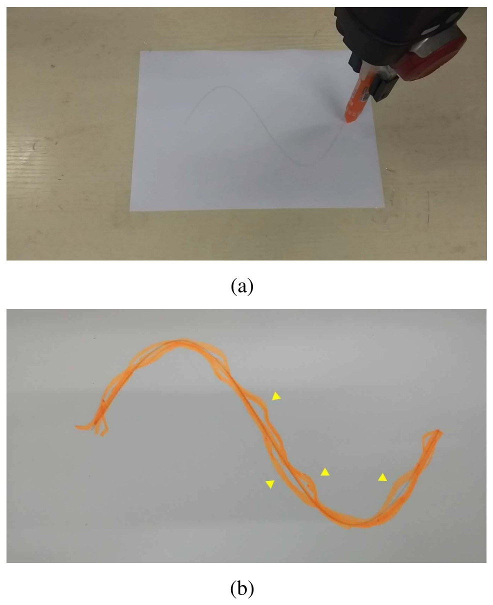


Fig. 13. (a) Experiment setup for the drawing task. (b) Demonstration trajectories of the drawing task.

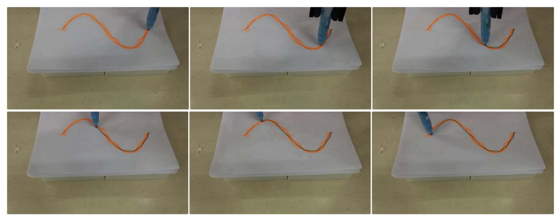
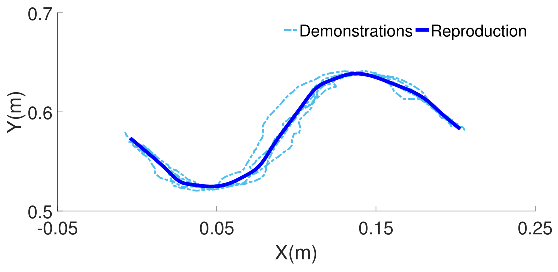


Fig. 14. Robot performs the drawing task with the learned motion.

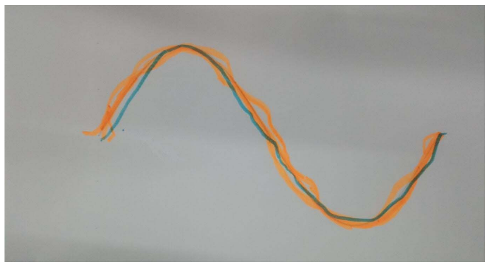
and the learning performance are tested. In these experiments, the demonstrations are performed by guiding the robot manipulator.

1) Ability of Generalization: In this experiment, the tutor demonstrates how to pour water into a cup placed on the table, as shown in Fig. 9. The demonstration process is repeated five times. The joints $w0$, $w1$, $s0$, and $e1$ are moved while the others are fixed. The joint angles are recorded and used for learning the modified DMP. The parameters of the DMP model are set as $\tau_s = 1$, $l_1 = 25$, $l_2 = 10$, and $\alpha_1 = \alpha_2 = 8$.

The learning results are shown in Fig. 10. The motions of the four joints are reproduced from the demonstrations, which synthesize the features of these demonstrations and enable the robot to complete the pouring task successfully. Subsequently, the target of the motion is modulated to the other cup. As shown in Fig. 11, the movement trajectory of each joint angle converges to the new goal and the profile of each reproduction is retained. We test the reproduced motion and the generalized motion on the robot. As shown in Fig. 12(a), the robot completes the pouring task successfully, and as shown in Fig. 12(b), the robot can pour water into the other cup.



(a)



(b)

Fig. 15. (a) Learning result using the DMP-based motion model in a drawing task. The motions are modeled in the task space. (b) Result of drawing. The blue curve is drawn by the robot after learning.

2) Learning Performance: To further validate the learning performance of the DMP-based motion model, we design a drawing task for the robot; the experimental setup is shown in Fig. 13(a). Here, the robot is required to draw an image of a sinusoid on paper after the tutor demonstrates the task five times. As shown in Fig. 13(b), the demonstrations are defective and the curves are irregular. One of the reasons is that the demonstrator is drawing on this paper indirectly by holding the robot's wrist, which affects the exertion of the drawing skill. The demonstrations are modeled in the task space, and the robot performs the drawing task after learning (Fig. 14). As shown in Fig. 15(a), a smooth curve is reproduced by the motion model given multiple demonstrations. We can also see that the recorded trajectories are distorted due to measurement errors of the sensors. As shown in Fig. 15(b), the curve that the robot draws is smoother than those of the demonstrations, thus validating the learning performance of the DMP-based motion model.

VI. Conclusion

In this paper, a novel robot learning system comprised of motion generation and trajectory tracking is developed. A DMP model is chosen as the basis of the motion model because of its generalization ability. To improve the learning performance, GMM and GMR are employed for estimating the nonlinear function of the motion model. With this modification, the model can extract more motion features from multiple demonstrations of a specific task and generate motions that synthesize these additional features. Besides, an NN-based controller is designed to overcome the impact of the unknown payload so that the manipulator is able to track the generated motions more accurately. Several experiments have been performed on the Baxter robot to test the performance of our proposed methods, which can be used to facilitate robot learning at a higher level. In the future work, we will further integrate the reinforcement learning into our system to improve the learning capacity of the robot.

References

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cognit. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2008, pp. 1371–1394.

[3] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[4] 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 Trans. Syst., Man, Cybern. A, Syst.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8002637/, doi: 10.1109/TSMC.2017.2705279.

[5] C. Yang, K. Huang, H. Cheng, Y. Li, and C.-Y. Su, "Haptic identification by ELM-controlled uncertain manipulator," *IEEE Trans. Syst., Man, Cybern. A, Syst.*, vol. 47, no. 8, pp. 2398–2409, Aug. 2017.

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

[7] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Robot. Res.*, vol. 32, no. 3, pp. 263–279, 2013.

[8] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1360–1370, Dec. 2012.

[9] Y. Zhao, R. Xiong, L. Fang, and X. Dai, "Generating a style-adaptive trajectory from multiple demonstrations," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 7, pp. 103–111, 2014.

[10] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *Proc. IEEE Int. Conf. Mach. Learn.*, 2008, pp. 144–151.

[11] S. Calinon and A. Billard, "Statistical learning by imitation of competing constraints in joint space and task space," *Adv. Robot.*, vol. 23, no. 15, pp. 2059–2076, 2009.

[12] S. Calinon, "Robot learning with task-parameterized generative models," in *Robotics Research*. Cham, Switzerland: Springer, 2018, pp. 111–126.

[13] L. Rozo, P. Jiménez, and C. Torras, "A robot learning from demonstration framework to perform force-based manipulation tasks," *Intell. Service Robot.*, vol. 6, no. 1, pp. 33–51, 2013.

[14] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 2, pp. 286–298, Apr. 2007.

[15] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Nov./Dec. 2012, pp. 323–329.

[16] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," in *Lazy Learning*. Dordrecht, The Netherlands: Springer, 1997, pp. 75–113.

[17] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Comput.*, vol. 17, no. 12, pp. 2602–2634, 2005.

[18] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009.

[19] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schäffer, and E. Burdet, "Human-like adaptation of force and impedance in stable and unstable interactions," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 918–930, Oct. 2011.

[20] C. Yang, Y. Jiang, Z. Li, W. He, and C.-Y. Su, "Neural control of bimanual robots with guaranteed global stability and motion precision," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1162–1171, Jun. 2017.

[21] L. Cheng, Y. Lin, Z.-G. Hou, M. Tan, J. Huang, and W. J. Zhang, "Integrated design of machine body and control algorithm for improving the robustness of a closed-chain five-bar machine," *IEEE/ASME Trans. Mechatronics*, vol. 17, no. 3, pp. 587–591, Jun. 2012.

[22] Y. J. Liu, S. C. Tong, D. Wang, T. S. Li, and C. L. P. Chen, "Adaptive neural output feedback controller design with reduced-order observer for a class of uncertain nonlinear SISO systems," *IEEE Trans. Neural Netw.*, vol. 22, no. 8, pp. 1328–1334, Aug. 2011.

[23] Z. Mao and F. Zhao, "Structure optimization of a vibration suppression device for underwater moored platforms using CFD and neural network," *Complexity*, vol. 2017, Dec. 2017, Art. no. 5392539. [Online]. Available: https://www.hindawi.com/journals/complexity/2017/5392539/abs/

[24] C. Yang, X. Wang, L. Cheng, and H. Ma, "Neural-learning-based telerobot control with guaranteed performance," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3148–3159, Oct. 2017.

[25] X. Yin and Q. Chen, "Learning nonlinear dynamical system for movement primitives," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2014, pp. 3761–3766.

[26] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 763–768.

[27] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 145–157, Feb. 2012.

[28] H. G. Sung, "Gaussian mixture regression and classification," Ph.D. dissertation, Rice Univ., Houston, TX, USA, 2004.

[29] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc., B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[31] J.-J. E. Slotine and L. Weiping, "Adaptive manipulator control: A case study," *IEEE Trans. Autom. Control*, vol. AC-33, no. 11, pp. 995–1003, Nov. 1988.

[32] C. Yang, X. Wang, Z. Li, Y. Li, and C.-Y. Su, "Teleoperation control based on combination of wave variable and neural networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 8, pp. 2125–2136, Aug. 2017.

[33] L. Cheng, Z.-G. Hou, M. Tan, and W.-J. Zhang, "Tracking control of a closed-chain five-bar robot with two degrees of freedom by integration of an approximation-based approach and mechanical design," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 5, pp. 1470–1479, Oct. 2012.

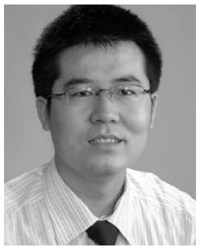
Chenguang Yang (M'10–SM'16) received the B.Eng. degree in measurement and control from Northwestern Polytechnical University, Xi'an, China, in 2005, and the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010.

He received postdoctoral training with Imperial College London, London, U.K. His current research interests include robotics and automation.

Dr. Yang was a recipient of the Best Paper Award in the IEEE TRANSACTIONS ON ROBOTICS and a number of international conferences.

Chuize Chen received the B.Eng. degree in automation from the South China University of Technology, Guangzhou, China, in 2017, where he is currently pursuing the M.S. degree..

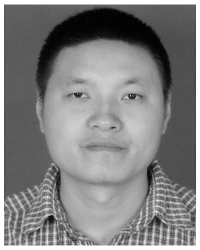
His current research interests include human–robot interaction, machine learning, and robotics.



Wei He (S'09–M'12–SM'16) received the B.Eng. and M.Eng. degrees from the College of Automation Science and Engineering, South China University of Technology, Guangzhou, China, in 2006 and 2008, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, in 2011.

He is currently a Full Professor with the School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing, China. He has co-authored two books published in Springer and authored or co-authored over 100 international journal and conference papers. His current research interests include robotics, distributed parameter systems, and intelligent control systems.

Dr. He is a member of the IFAC TC on Distributed Parameter Systems, the IFAC TC on Computational Intelligence in Control, and the IEEE CSS TC on Distributed Parameter Systems. He was a recipient of the Newton Advanced Fellowship from the Royal Society Award, U.K., and the IEEE SMC Society Andrew P. Sage Best Transactions Paper Award in 2017. He serves as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, and an Editor for the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, *Neurocomputing*, and the *Journal of Intelligent and Robotic Systems*.



Rongxin Cui (M'09) received the B.Eng. and Ph.D. degrees from Northwestern Polytechnical University, Xi'an, China, in 2003 and 2008, respectively.

From 2008 to 2010, he was a Research Fellow with the Centre for Offshore Research and Engineering, National University of Singapore, Singapore. He is currently a Professor with the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China. His current research interests include control of nonlinear systems, cooperative path planning and control for multiple robots, control and navigation for underwater vehicles, and system development.

Dr. Cui serves as an Editor for the *Journal of Intelligent and Robotic Systems*.

Zhijun Li (M'07–SM'09) received the Ph.D. degree in mechatronics from Shanghai Jiao Tong University, Shanghai, China, in 2002.

From 2003 to 2005, he was a Post-Doctoral Fellow with the Department of Mechanical Engineering and Intelligent Systems, University of Electro-Communications, Tokyo, Japan. From 2005 to 2006, he was a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, and Nanyang Technological University, Singapore. From 2012 to 2017, he was a Professor with the College of Automation Science and Engineering, South China University of Technology, Guangzhou, China. Since 2017, he has been a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. His current research interests include service robotics, teleoperation systems, nonlinear control, and neural network optimization.