Robot learning system based on Dynamic Movement Primitives and Neural Network

Ying Zhang

College of Automation Science and Engineering, South China University of Technology, Guangzhou, China.

Miao Li

The Institute of Technological Sciences, Wuhan University, Wuhan, China.

Chenguang Yang

Bristol Robotics Laboratory, University of the West of England, Bristol, BS16 1QY, UK.

Abstract

In the process of Human-robot skill transfer, we require the robot to reproduce the trajectory of teacher and expect that the robot can generalize the learned trajectory. For the trajectory after generalization, we expect that the robot arm can accurately track. However, because the model of the robot can not be accurately obtained, some researchers have proposed using a neural network to approximate the unknown term. The parameters of the traditional RBF neural network are usually selected through the empirical and trial-and-error method, which maybe biased and inefficient. In addition, due to the end-effector of the mechanical arm trajectory will be constantly changing according to the needs of the task, when the neural network of compact set cannot contain the whole input vector, the neural network cannot achieve the ideal approximation effect. In this paper, the broad neural network is used to approximate the unknown terms of the robot. This method can reuse the motion controller that has been learned and complete other motions in the robot operating space without relearning its weight parameters. In this paper, the effectiveness of the proposed method is proved by the ultrasound scanning task.

Keywords: Adaptive neural control; Broad learning; Dynamic movement primitive; Learning from demonstration; Human-robot skill transfer; Force control

1. Introduction

In daily life and work, human can easily learn some basic task skills. They can flexibly expand these basic skills to other more complex tasks through combination, optimization and other ways due to human's excellent learning, summary and expansion ability. There have been efforts to give robots this kind of intelligent behavior. Man-machine skill transfer can transfer human skills to robots, so that robots can replace people to perform some repetitive and tiring tasks. Doing ultrasound scanning tasks for a long time can easily cause the doctors' muscles too tired. Besides, the doctor needs to contact the patient in the process of ultrasound task, which can lead to the spread of disease. In this paper, the robot arm is considered to perform B-ultrasound scanning instead of the doctor, to reduce the muscle strain of the doctor and improve the work efficiency. Meanwhile, the isolation between the doctor and the patient will increase safety.

The commonly used skills representation models include the dynamic movement primitives (DMPs) and probabilistic models, such as the Gaussian Mixture Model (GMM), Hidden Markov model (HMM) and Hidden Semi-Markov Model (HSMM). The dynamic motion primitive model is essentially a second-order nonlinear system (spring-damping system) to approximate a motion trajectory. Dynamic motion primitive framework [1, 2] and its derivative structure [3] have been proposed by many researchers and widely applied to motion modeling based on imitative learning [4] and reinforcement learning [5]. [6] improved the DMP method and proposed the DMP+ algorithm. On the basis of retaining the advantages of DMPS, it can obtain lower mean square error in the case of the same kernel function, and at the same time allows the learning trajectory to be effectively modified by updating the subset of the kernel. However, DMPs can only learn one demonstration. Even for professionals, it is difficult to obtain a good trajectory only through one demonstration. In this paper, DMPs are combined with GMM and the Gaussian mixture regression (GMR) to solve the shortcoming that the original DMPs can only learn from one demonstration. Some scholars have developed several variant models based on the original GMM. For example, the work [7] proposed an Incremental, Local and Online variation of Gaussian Mixture Regression (ILO-GMR) model, which enables the robots to learn new skills online, thus avoiding repeat-

*Corresponding author

Email address: cyang@ieee.org (Chenguang Yang)

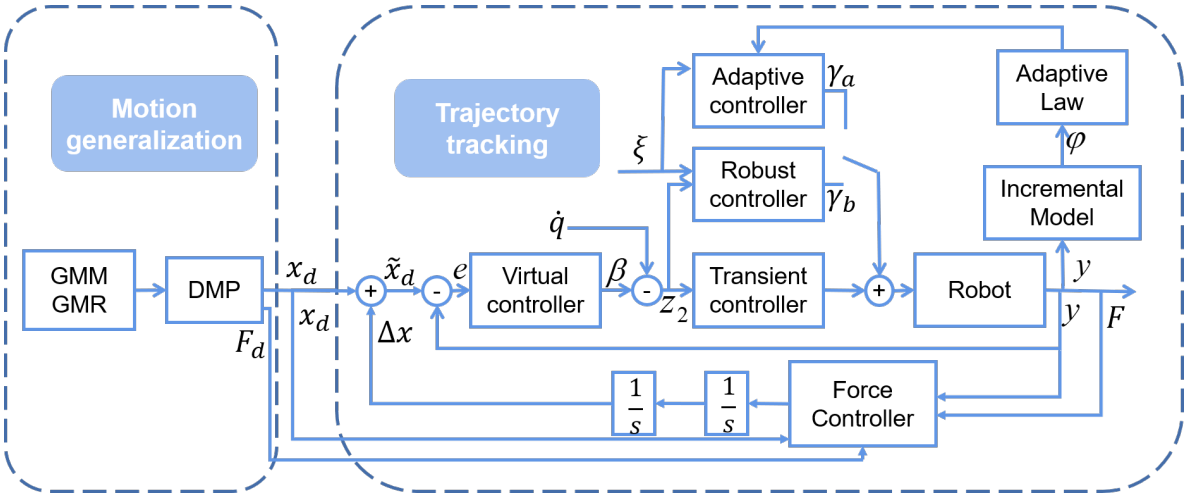


Figure 1: The overview of the proposed approach.

ed adjustment of model parameters and improving skill learning efficiency. A task-Parameterized Gaussian Mixture Model (TP-GMM) is proposed in [8] to integrate task information parameters into the learning process of the model, so that it can have a certain perception effect on the external environment in the task repetition stage. The article [9] further optimized the TP-GMM model and reduced the dependence of the model on external environment information.

The simulation performance of the robot also depends on the accuracy of the trajectory tracking controller involved in robot dynamics. In general, model-based controls perform better if the model is accurate enough [10]. However, it is impossible to obtain an accurate motion model because of the uncertainty of load and unknown disturbance. Approximation-based controllers are designed to overcome these uncertainties [11, 12, 13]. [14] propose a dynamic neural network discriminator (DNN) based on Lyapunov function to approximate the unknown dynamics of the system. The neural network has been widely used in controller design due to its powerful approximation capability [15, 16]. In the design of a traditional RBF neural network, the selection of its center point and width values determines the performance of the neural network, and these parameters are usually selected through the designer's experience and trial-and-error method, which is both biased and low efficiency [17, 18]. For the traditional neural network, when the input vector is input into the neural network, the neural nodes near the input vector will be activated due to the characteristics of Gaussian kernel, to achieve the purpose of learning. However, the neural nodes far away from the input vector will not realize the function of learning. If the compact set of the neural network cannot contain the entire input vector, the neural network cannot achieve the ideal approximation effect. In this paper, the broad neural network is used to solve this problem. When the width radial neural network is initialized, the neural nodes of the network take the initial state of the system as the first node. Subsequently, the nodes of the neural network will determine whether new network nodes should be added according to the change of the input vector of the system and the position relationship of the existing nodes in the network. When there is no need to add network nodes, the learning process is consistent with the learning principle of traditional radial neural network. When the input vector is far away from the original system network nodes, that is, beyond the compact set domain of the neural network, it is necessary to increase the network nodes, so that the input vector can be within the expanded compact set domain, to ensure the approximation performance of the neural network [17].

However, considering position control alone cannot achieve tasks which requires force constraints. A survey [19] designed a cost function with muscle activation parameters as the variable and deduced the updating rules of impedance and feedforward force according to the position error by gradient descent method. The algorithm was verified on a single-degree-of-freedom platform. The work [20] improved the algorithm by considering motion position error and force trajectory error in the parameters, and integrating the algorithm and motion planning into a teaching system. Finally, the method was verified on the virtual robot platform in the simulation environment, and the effectiveness of the algorithm was proved. In this paper, the learning of force information is added on the basis of learning the position information, and the force controller is added to realize the force tracking.

The structure diagram of this paper is shown in Fig. 3, including the trajectory generation part and the trajectory tracking part. Firstly, GMM and GMR were used to fit the teaching data, and then DMPs was used to model the data after fitting. In this paper, a neural network controller based on broad learning is used to track the desired trajectory generated by DMPs. Finally, the effectiveness of the proposed framework is proved by experiments.

The rest of the article structure is as follows: In Section 2, the method of trajectory generation and the force controller used in this paper are introduced. In Section 3, we mainly introduce the principle of neural network controller based on broad learning and proves its stability. The effectiveness of the framework proposed in this paper is verified in Section 4. Finally, Section 5

summarizes the whole paper.

2. Basic Model of Discrete Motion

2.1. Dynamic Movement Primitives (DMPs)

In this paper, motion DMPs and force DMPs can be obtained by using DMPs model to fit motion trajectory and force trajectory respectively. The principles of motion DMPs and force DMPs used in this paper are stated as follows:

2.1.1. DMPs for motion trajectory

The essence of DMPs is a second-order nonlinear dynamical system including spring and damper. A single degree of freedom motion can be expressed by the following formula [21]:

$$\tau\dot{\beta_2} = a(g - \beta_1) - b\beta_2 + f(s;\omega) \tag{1}$$

$$\tau\dot{\beta_1} = \beta_2 \tag{2}$$

$$\tau\dot{s} = -k_1 s \tag{3}$$

where, we ignore the time variable for the sake of simplicity, for example, $\beta_1(t)$ is represented by β_1; a and b represent the damping coefficient and spring constant of the system respectively, and a is usually set as $a = b^2/4$; g is the target value of the motion trajectory; τ represents the time scaling constant; β_1 and β_2 represent the position and velocity of motion trajectories respectively, and the relationship between these two variables is shown in formula (2); w means the weight of the Gaussian model, and s is the phase variable of the system, which is calculated by the regular system of equation (3), where k_1 is a positive constant. The nonlinear function $f(s;\omega)$ is defined as:

$$f(s;\omega) = \frac{\sum_{i=1}^{N}\phi_i\omega_i}{\sum_{i=1}^{N}\phi_i}(g - \beta_0)s \tag{4}$$

$$\phi_i = exp(-d_i(s - c_i)^2) \tag{5}$$

where, c_i, d_i and w_i are the center, width and weight of the i-th kernel function respectively; β_0 is the initial value of the motion trajectory; N is the total amount of Gaussian models. In general, the initial value of s is set to 1, which gradually decays to zero. Since the value of s tends to zero, the nonlinear function $f(s;\omega)$ is bounded, and the model becomes a stable second order spring-damped system.

In general, supervised learning algorithms such as local weighted regression algorithm (LWR) is used to determine model parameters ω. Given the teaching trajectory $\beta(t)$, where $t = [1, 2..., T]$, $g = \beta(T)$, the objective function can be determined according to formula (1):

$$f_{target} = \tau\dot{\beta_2} - K(g - \beta_1) + D\beta_2 \tag{6}$$

w can be determined by the following formula:

$$\arg\min_{\omega} \sum (f_{target} - f(s;\omega))^2 \tag{7}$$

2.1.2. DMPs for force trajectory

Similar to the DMPs for motion trajectory planning, force trajectories obtained by teaching are modeled by DMPs. Given the trajectory of teaching force $f(t)$, where $t = [1, 2, ..., T]$, $g = f(T)$:

$$\tau\dot{f_2} = c(f_g - f_1) - df_2 + f(s;\gamma) \tag{8}$$

$$\tau\dot{f_1} = f_2 \tag{9}$$

$$f(s;\gamma) = \frac{\sum_{i=1}^{N}\phi_i\gamma_i}{\sum_{i=1}^{N}\phi_i}(f_g - f_0)s \tag{10}$$

where, c and d respectively represent the damping coefficient and spring constant of the system, and c is usually set as $c = d^2/4$; f_g and f_0 are the target value and initial value of the force trajectory respectively; τ represents the time scaling constant; f_1 and f_2 respectively represent the position and velocity of force trajectories, and the relationship between these two variables is shown in formula (9); γ_i means the weight of the i-th Gaussian model. The two transformation systems are driven by the same phase variable s, so that they can be synchronized on the time axis. The estimation process of model parameters of force DMPs is the same as that of motion DMPs, as shown in formula (6) and (7).

2.2. Learning from Multiple Demonstrations

For the same task, it is usually necessary to teach the robot several times in order to learn the human movement characteristics better. Accordingly, multiple teaching trajectories need to be modeled in order to learn skill representation from the results of multiple teaching better. The main methods can be divided into two categories:

2.2.1. Subjective evaluation

In this method, the results of each teaching are evaluated by the quality and performance of the teaching, and then a comprehensive result is obtained by some weighted method. For example, in the retransport object experiment, the teaching quality is evaluated according to the smoothness of each transport object process, whether it leads to a large interaction force and other factors, and accordingly higher weight is given to the teaching data with high teaching quality. In order to learn from the results of multiple demonstrations, we can take a direct approach to determining the weight of each session. This evaluation method is simple and intuitive, but highly subjective.

In order to learn from the results of multiple teachings, we can adopt a direct way to determine the weight of each teaching. First, score them based on the performance of each teaching. Assign a coefficient k_i related to the score π_i for each teaching, that is:

$$k_i = \frac{\pi_i}{\sum_{i=1}^{L}\pi_i} \tag{11}$$

where, L represents the total teaching number.

Then, the final teaching trajectory y can be obtained according to the weighted average of each teaching output y_i:

$$y = \sum_{i=1}^{L} k_i y_i \tag{12}$$

2.2.2. Method based on GMM model

First, a joint probability distribution $P(t, \theta)$ was constructed to represent the demonstration trajectory $(t_n, \theta_{n,k})|n = (1, 2, ..., N), k = (1, 2, ..., K)$ based on N demonstration data, where N stands for teaching time and K stands for K times of demonstration. $P(t, \theta)$ is defined as follows [22]:

$$P(t, \theta) = \sum_{l=1}^{L} \pi_l H(t, \theta; \mu_l, \Sigma_l) \tag{13}$$

In the formula, $\pi_l \in [0, 1]$ is the prior coefficient and satisfies:

$$\sum_{l=1}^{L} \pi_l = 1 \tag{14}$$

The l-th ($l = 1, ..., L$) center parameter μ_l and covariance matrix Σ_l are defined respectively as:

$$\mu_l = \begin{bmatrix} \mu_t^{(l)} \\ \mu_\theta^{(l)} \end{bmatrix}, \Sigma_l = \begin{bmatrix} \Sigma_{t,t}^{(l)}, \Sigma_{t,\theta}^{(l)} \\ \Sigma_{\theta,t}^{(l)}, \Sigma_{\theta,\theta}^{(l)} \end{bmatrix} \tag{15}$$

where $\Sigma_{t,t}^{(l)}$ represents the variance of t of the l-th Gaussian component, and $\Sigma_{t,\theta}^{(l)}$ represents the covariance of t and θ of the l-th Gaussian component. Gaussian probability distribution $H(t, \theta; \mu_l, \Sigma_l)$ has the following forms:

$$\begin{aligned} H(t, \theta; \mu_l, \Sigma_l) = (2\pi_l \sqrt{|\Sigma_l|})^{-1} * \\ exp(-\frac{1}{2}([t, \theta]^T (\Sigma_l)^{-1} ([t, \theta]^T - \mu_l))) \end{aligned} \tag{16}$$

where, model parameters π_l, μ_l, and Σ_l can be determined by using Expectation-Maximization (E-M) algorithm. Because the algorithm is very sensitive to the initial values of parameters, k-means algorithm can be used to initialize E-M model parameters. The algorithm divides data sets into multiple sets in order to find a partitioned set $F = (F_1, F_2, ..., F_K)$ and minimizing the sum of the square deviation of each set [23].

$$\hat{F} = \underset{F}{argmin} \sum_{k=1}^{K} \sum_{x \in F_k} \|x - \bar{x}_k\|_2 \tag{17}$$

where, $x = [t_n, \theta_{n,k}]^T \in R^{2\times1}$, $\bar{x}_k \in R^{2\times1}$ is the average of set F_k. Then E-M algorithm is used to find the best GMM parameters. The main process is to maximize the logarithmic likelihood function by finding the parameters $\hat{\pi}_k, \hat{\mu}_k$ and $\hat{\Sigma}_k$ [21]:

$$\left(\hat{\pi}_k, \hat{\mu}_k, \hat{\Sigma}_k\right) = \underset{\pi_k, \mu_k, \Sigma_k}{\arg\max} \log\left(p\left(t, \theta \mid \pi_k, \mu_k, \Sigma_k\right)\right) \tag{18}$$

in addition, the number of Gaussian models also needs to be determined in advance, which can be selected empirically according to the complexity of trajectory shape.

Then, the Gaussian Mixture Regression (GMR) algorithm is used to estimate the function $\theta(t)$. According to the reference [24], formula (13) can be written as:

$$P(t, \theta) = \sum_{k=1}^{K} \pi_k \mathrm{H}\left(\theta \mid t; m_k(t), \sigma_k^2\right) \mathrm{H}\left(t; \mu_t^{(k)}, \Sigma_{t,t}^{(k))}\right) \tag{19}$$

where,

$$m_k(t) = \mu_\theta^k + \left(\Sigma_{\theta,t}^{(k)}\right)^{-1} \left(t - \mu_t^{(k)}\right) \tag{20}$$

$$\sigma_{(k)}^2 = \Sigma_{k,\theta,\theta} - \Sigma_{\theta,t}^{(k)} (\Sigma_{t,t}^{(k)})^{-1} \Sigma_{t,\theta}^{(k)} \tag{21}$$

The marginal density of t is:

$$P(t) = \int P(t, \theta) d\theta = \sum_{k=1}^{K} \pi_k \mathrm{H}\left(t_k; \mu_t^{(k)}, \Sigma_{t,t}^{(k)}\right) \tag{22}$$

and the conditional probability distribution function of $\theta|t$ is:

$$P(\theta \mid t) = \sum_{k=1}^{K} w_k(t) H\left(\theta, m_k(t), \sigma_k^2\right) \tag{23}$$

where,

$$w_k(t) = \frac{\pi_k \mathrm{H}\left(t; \mu_t^{(k)}, \Sigma_{t,t}^{(k))}\right)}{\sum_{k=1}^{K} \pi_k H\left(t; \mu_t^{(k)}, \Sigma_{t,t}^{(k)}\right)} \tag{24}$$

Then, the estimated value of function $\theta(t)$ is:

$$\theta(t) = \sum_{k=1}^{K} w_k m_k(t) \tag{25}$$

2.3. Force Control

During the execution of ultrasound scanning, we need to control the force applied to the skin to avoid discomfort and to ensure the quality of the ultrasound image. The force received by the object in the reproduction process is hoped to be maintained near the expected value. In this paper, a force controller is added on the basis of the position controller to realize the force control in the ultrasound process. The force controller used in this paper is:

$$\Delta\ddot{X} = m\left(X_d - X\right) - n\dot{X} - \left(F_d - F\right) \tag{26}$$

where,

$$m = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & m_z \end{bmatrix}, n = \begin{bmatrix} n_x & 0 & 0 \\ 0 & n_y & 0 \\ 0 & 0 & n_z \end{bmatrix} \tag{27}$$

represent damping and stiffness respectively. By adjusting these two parameters, the force tracking performance of the robot terminal can be changed. $X_d = [x_d, y_d, z_d]^T \in R^{3\times1}$ and $X = [x, y, z]^T \in R^{3\times1}$ are the expected trajectory and the actual trajectory respectively; $F_d = [f_{d,x}, f_{d,y}, f_{d,z}] \in R^{3\times1}$ and $F = [f_x, f_y, f_z]^T \in R^{3\times1}$ are the expected force and the actual force respectively. The output of the force controller is acceleration $\Delta\ddot{X} = [\Delta\ddot{x}, \Delta\ddot{y}, \Delta\ddot{z}]^T \in R^{3\times1}$, and by integrating $\ddot{x}$ twice and adding it to the desired position x_d, the force can be adjusted as shown in Fig. 3.

3. RBFNN With Broad Learning Framework

3.1. Dynamics Description

According to the Lagrange-Euler form, the dynamics model of robot can be described as [25]:

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + F_c(\dot{\theta}) + G(\theta) = \tau_c \tag{28}$$

where, $M(\theta) \in R^{N\times N}$ is the inertia matrix of the manipulator, the matrix is symmetric and positive definite [22]. $C(\theta,\dot{\theta}) \in R^{N\times N}$ represents the centrifugal force and the Coriolis force vectors. The inertial matrix of the manipulator is symmetric and positive definite, and the inertial matrix and Coriolis force vector satisfy the oblique symmetric relation: $x^T\left[\dot{M}_x(\theta) - 2C_x(\theta,\dot{\theta})\right]x = 0$. $F_c(\dot{\theta}) \in R^{N\times N}$ is the Coulomb friction term, $G(\theta) \in R^{N\times 1}$ is the gravity term. $\tau_c = \tau - \tau_p$, τ_c and τ_p represent the control moment and load moment respectively.

3.2. Radial Basis Neural Network (RBFNN)

Due to the uncertainty of the environment and the limitation of measuring tool accuracy, it is difficult to obtain the precise model parameters of the robot. Therefore, the function approximation adaptive control method based on neural network is widely used, that is, the radial basis function neural network is used to approximate the robot indeterminate term. The general form of radial basis function is as follows:

$$S(X-\psi) = S(\|X-\psi\|) \tag{29}$$

where X represents the input vector of the neural network, $\|X-\psi\|$ represents the standard Euclidean distance between vector X and center point vector ψ. It can be seen that the output of the radial basis function depends on the distance between the input and the center point. Gaussian function is a common form of radial basis function neural network control due to its advantages of good smoothness, simple form of expression and differentiability of any order. Its expression is as follows:

$$S\left(\|x-\varphi_i\|\right) = \exp\left(-\frac{\|x-\varphi_i\|^2}{\eta_i}\right) \tag{30}$$

where, η_i represents the adjustable width of the i-th Gaussian kernel function, and φ_i represents the center point of the i-th Gaussian kernel. Studies show [26] that RBF neural network can approximate any nonlinear continuous function with any accuracy as long as there are enough ganglia:

$$f^*(X) = W^T S(X) + \varepsilon(X) \quad \forall X \in \Omega_X \tag{31}$$

where W represents the weight of the neural network, $\Omega_X \in R^D$ stands for neural network emergency, $\varepsilon(X)$ represents the approximation error of the neural network, where for any $\varepsilon^* > 0$ there exists $|\varepsilon(X)| < \varepsilon^*$. Thus, the optimal estimation weight of the neural network can be expressed as:

$$W^* = \arg\min_{\widehat{W}\in\Omega_F}\left[\sup|\hat{f}(X \mid \widehat{W}) - f^*(X)|\right] \tag{32}$$

where, $X \in \Omega_X$, $\widehat{W}$ represents the estimated weight of the neural network, and Ω_F represents the estimated compact set of $\widehat{W}$.

3.3. RBFNN with Broad Learning System

We define the parameter vector of the newly added neural network node as [17]:

$$N = <\varphi_{\text{new}}, \eta_{\text{new}}, W_{\text{new}}> \tag{33}$$

where, φ_{new}, η_{new}, and W_{new} represent the newly added center point, width value and weight value of the radial basis function respectively.The distance between the input state x(t) and the neural network node is defined as:

$$\varphi_{\text{new}} = \bar{C}_{\min} + \beta\left(X(t) - \bar{C}_{\min}\right) \tag{34}$$

where, $\beta \in R$ is the adjustable parameter, which determines the distance between the location of the new node and the original neural network node set. $\bar{C}_{\min}$ represents the average node of network node set $C_{\min}$:

$$\bar{C}_{\min} = \frac{\sum_{i=1}^{k} c_{\min}(k)}{k}, \tag{35}$$

therefore, the set of center points updated after each calculation period t can be expressed as,

$$\varphi(t+T) = \begin{cases} [\varphi(t)\varphi_{\text{new}}] & \left\|X(t) - \bar{C}_{\min}\right\| > \varepsilon \\ \varphi(t) & \left\|X(t) - \bar{C}_{\min}\right\| \le \varepsilon \end{cases} \tag{36}$$

where, ε represents the adjustable threshold parameter of neural network node expansion. When the distance between the input state of the system and the average node of the neural network is greater than a certain threshold value, we can consider that the system input exceeds the virtual compact set of the neural network. In order to ensure the approximation accuracy of the neural network, a new network node needs to be added. On the contrary, the weights of the original neural network nodes need to be updated.

In addition, in order to meet the real-time requirements of robot control, the triangular expansion method is used in this paper to generate network enhancement nodes [27]. Assume k is the number of ganglion points of the radial basis neural network of current width, then the enhancement node can be expressed as:

$$H(t) = [H_1, H_2, \ldots, H_3] \tag{37}$$

where, $H_i = [\cos(s_i(X(t))), \sin(s_i(X(t)))], i = 1, \cdots k$; $s_i(\cdot)$ represents the kernel function of the i-th neural network node. Thus, the updated basis function and enhanced node vector of the width neural network are:

$$\begin{aligned} S_n(t+T) &= [S(t) \mid S_{ex}(t)] \\ &= \left[S(t) \mid \exp\left(-\|x-\varphi_{\text{new}}\|^2/\eta_{\text{new}}\right)\right] \\ H_n(t+T) &= [H(t) \mid H_{ex}(t)] \\ &= [H(t)|\cos(S_{ex}(t))|\sin(S_{ex}(t))] \\ W_n(t+T) &= [W(t) \mid W_{ex}(t)] \end{aligned} \tag{38}$$

Thus, the width radial basis neural network can be expressed as:

$$Y(t) = [S_n(t) \mid H_n(t)]\, W^T \tag{39}$$

3.4. Globally Stable Robot Broad Neural Network Tracking Control

After the design of the predetermined performance function is completed, a globally stable broad neural network controller is designed based on Obstacle Lyapunov function and backstepping method in this section. For the manipulator with degree of freedom, we first define its asymmetric obstacle Lyapunov function as [17]:

$$\begin{aligned} V_1 =& \frac{1}{2} z_1^T P^T z_1 + \sum_{i=1}^{N_d} \left(\frac{h_i(z_1)}{2} \ln \frac{1}{1-\delta_{b,i}^2} \right) \\ &+ \sum_{i=1}^{N_d} \left(\frac{1-h_i(z_1)}{2} \ln \frac{1}{1-\delta_{a,i}^2} \right) \end{aligned} \tag{40}$$

where, $\delta_{a,i} = \frac{z_1(i)}{\phi_{a,i}} = \frac{z_1(i)}{-k_{a,i} v_i(t)}$ $\delta_{b,i} = \frac{z_1(i)}{\phi_{b,i}} = \frac{z_1(i)}{-k_{b,i} v_i(t)}$, $k_{a,i}$ and $k_{b,i}$ are constants, $h_i(z_1)$ is defined as:

$$h_i(z_1) = \begin{cases} 1 & z_1 \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

The derivative of (33) with respect to t can be obtained as follows:

$$\dot{V}_1 = P^T \dot{z}_1 + \sum_{i=1}^{N_d} \left(\frac{(1-h_i)\delta_{a,i}^2}{1-\delta_{a,i}^2} + \frac{h_i \delta_{b,i}^2}{1-\delta_{b,i}^2} \frac{\dot{\varphi}_{b,i}}{\varphi_{b,i}} \right) \tag{42}$$

where, $P = [p(1), p(2), \cdots, p(N_d)]$, $p(i) = \frac{\chi_i^2}{(1-x_i^2) z_1(i)}$, $\chi_i = h_i \delta_{b,i} + (1-h_i)\delta_{a,i}$ represents the transient control vector. Thus, the virtual vacancy rate α can be designed as

$$\alpha = J^{\dagger}(\theta)(\dot{x}_d - K_1 z_1 - L(t) z_1) \tag{43}$$

where, $J^{\dagger} = J^T \left(J J^T \right)^{-1}$ represents the generalized inverse of the Jacobian matrix $J(\theta)$, $l_i(t) = \sqrt{\left(\frac{\dot{\phi}_{a,i}}{\phi_{a,i}} \right)^2 + \left(\frac{\dot{\phi}_{b,i}}{\phi_{b,i}} \right)^2 + c_i}$. c_i is a tunable normal number whose function is to guarantee the boundedness α go to zero. Substituting α into Equation (42), we can get:

$$\begin{aligned} \dot{V}_1 =& P^T J(\theta) Z_2 - P^T (K_1 z_1 + L(t) z_1) + \\ & \sum_{i=1}^{N_d} \left(\frac{(1-h_i)\delta_{a,i}^2}{1-\delta_{a,i}^2} + \frac{h_i \delta_{b,i}^2}{1-\delta_{b,i}^2} \frac{\dot{\varphi}_{b,i}}{\varphi_{b,i}} \right) \end{aligned} \tag{44}$$

Consider inequality $l_i(t) - h_i \frac{\phi_{a,i}}{\phi_{a,i}} - (1-h_i) \frac{\phi_{b,i}}{\phi_{b,i}} \geq 0$. Combining (36) and (37), we can get:

$$\dot{V}_1 \leq - \sum_{i=1}^{N_d} k_1(i) \frac{\chi_i^2}{\left(1-x_i^2\right)} + P^T J(q) z_2 \tag{45}$$

where, $\chi_i = h_i \delta_{b,i} + (1-h_i)\delta_{a,i}$.

Then, we define a positive definite Lyapunov function:

$$V_2 = V_1 + \frac{1}{2} z_2^T M(\theta) z_2 \tag{46}$$

$$\begin{aligned} \dot{V}_2 &= \dot{V}_1 + \frac{1}{2} z_2^T \dot{M}(\theta) z_2 + z_2^T M(\theta) \dot{z}_2 \\ &= \dot{V}_1 + z_2^T (\tau - G(\theta) - C(\theta, \dot{\theta})\alpha - \mathcal{M}(\theta)\dot{\alpha}) \\ &= \dot{V}_1 + z_2^T (\tau + F(\xi)) \end{aligned} \tag{47}$$

where, $\epsilon = [\theta, \dot{\theta}, \alpha, \dot{\alpha}]$. Since the robot arm models $M(\theta), C(\theta, \dot{\theta})$ and $G(\theta)$ are difficult to be accurately obtained, $F(\xi)$ is the approximation term representing the robot model, which will be estimated by the width radial basis neural network discussed above.

Because the input of the neural network needs to be defined in the compact set domain Ω_0 of the neural network, the approximation ability of the neural network and the stability of its controller can be guaranteed. For the problem of global consistency stability of the controller, the control algorithm of switching function is proposed in [28]. In order to achieve the dynamic global consistency stability of the controller, we define a smooth switching function:

$$\bar{E}(\xi) = \operatorname{diag}(E_1(\xi), E_2(\xi), \cdots, E_{N_d}(\xi)) \tag{48}$$

where, $E_i(\xi) = \prod_{k=1}^{4} e_k(\xi(N_d(k-1)+i))$, and $e_k(\xi)$ is defined as,

$$e_k(\xi) = \begin{cases} 1 & |\xi| < d_a \\ \frac{d_b^2 - \xi^2}{d_b^2 - d_a^2} \exp\left(\frac{\xi^2 - d_a^2}{\sigma_i (d_b^2 - d_a^2)} \right)^2 & \text{others} \\ 0 & |\xi| > d_b \end{cases} \tag{49}$$

where, $d_a = \varrho_a * q_f$, $d_b = \varrho_b * q_f$, q_f represents the radius corresponding to the center point farthest from the remote point in the neural network, and ϱ_a and ϱ_b represent the adjustable distance threshold. Then, the globally uniformly stable adaptive incremental neural network controller can be designed as:

$$\tau_x = -K_2 z_2 - P^T J(\theta) - \bar{E}(\xi)\Upsilon_a - (1 - \bar{E}(\xi))\Upsilon_b \tag{50}$$

where K_2 is a positive definite vector, Υ_a and Υ_b correspond to the neural network controller and the robust controller in Fig. 1 respectively, and their form is defined as follows:

$$\Upsilon_a = \hat{Y}(\xi) \tag{51}$$

$$\Upsilon_b = UR\left(\frac{U z_2}{\sigma} \right) \tag{52}$$

where, $\hat{Y}(\xi)$ is the output of the width neural network controller, U and σ is an adjustable normal number. $R(\cdot) = [r_1, \cdots, r_{N_d}]$ and $r_i \left(\frac{u_i z_2}{\varpi_i} \right) = \tanh\left(\frac{u_i z_2(i)}{\varpi_i} \right)$. The weight update low of the neural network (32) is designed as:

$$\dot{\widehat{W}}_n = \Lambda \left(\bar{E}(\xi) z_2 S_n(\xi) - \beta \widehat{W}_n \right) \tag{53}$$

where Λ and β are positive definite matrices. $S_n(\xi)$ represents the kernel function of the n-th neural network node.

3.5. Stability Proof

According to equation (47), a Lyapunov function is designed [29, 30]:

$$V = V_2 + \frac{1}{2}\sum_{i=1}^{N_i} \widetilde{W}_{n_i}^T \Lambda^{-1} \widetilde{W}_{n_i} \tag{54}$$

where, Λ is a positive definite matrix, and the derivative of (54) according to time t is:

$$\begin{aligned}
\dot{V} &= \dot{V}_2 + \sum_{i=1}^{N_i} \widetilde{W}_{n_i}^T \Lambda^{-1} \dot{\widetilde{W}}_{n_i} \\
&= \dot{V}_1 + z_2^T(\tau + F(\xi)) + \sum_{i=1}^{N_i} \widetilde{W}_{n_i}^T \Lambda^{-1} \dot{\widetilde{W}}_{n_i} \\
&= \dot{V}_1 - K_2 z_2^T z_2 - P^T J(\theta) z_2^T \\
&+ \sum_{i=1}^{N_i} z_2(i) \left[-\bar{E}_i \left(\widehat{W}_{n_i}^T - W_{n_i}^{*T} \right) S_{n_i}(\xi) \right. \\
&\left. + \bar{E}_i \widehat{W}_{n_i}^T S_{n_i}(\xi) + \epsilon(i) \right] \\
&+ \sum_{i=1}^{N_i} z_2(i) \left(1 - \bar{E}_i\right) \left(F_i - u_i \tanh\left(\frac{u_i z_2(i)}{\varpi_i} \right) \right) \\
&+ \sum_{i=1}^{N_i} \left[-\beta \widetilde{W}_{n_i}^T \left(W_{n_i}^* + \widetilde{W}_{n_i} \right) \right]
\end{aligned} \tag{55}$$

according to Young's inequality, we can obtain the following inequality

$$F_i z_2(i) - u_i z_2(i) \tanh\left(\frac{u_i z_2(i)}{\varpi_i} \right) \leq \iota \varpi_i \tag{56}$$

thus, (55) can be changed into:

$$\begin{aligned}
\dot{V} &\leq \sum_{i=1}^{N_d} \left[-K_1(i) \frac{\chi_i^2}{\left(1 - \chi_i^2\right)} \right] \\
&+ \sum_{i=1}^{N_i} \left[-\left(K_2(i) - \frac{1}{2} \right) z_2^2(i) - \frac{1}{2}\beta(i) \left\| \begin{array}{c} \widetilde{W}_i \\ \widetilde{W}_{ex_i} \end{array} \right\|^2 \right] \\
&+ \sum_{i=1}^{N_i} \left[\frac{1}{2}\beta(i) \left\| \begin{array}{c} W_i^* \\ W_{ex_i}^* \end{array} \right\|^2 + \frac{1}{2}\epsilon^2(i) + \iota \varpi_i \right]
\end{aligned} \tag{57}$$

consider the following inequality:

$$-\frac{\chi_i^2}{\left(1 - \chi_i^2\right)} \leq -\ln \frac{1}{\left(1 - \chi_i^2\right)} \quad \forall |\chi_i| < 1 \tag{58}$$

(57) can be changed into:

$$\begin{aligned}
\dot{V} &\leq \sum_{i=1}^{N_d} \left[-K_1(i) \ln \frac{1}{\left(1 - \chi_i^2\right)} \right] \\
&+ \sum_{i=1}^{N_i} \left[-K_3(i) z_2^2(i) - \frac{1}{2}\beta(i) \left\| \widetilde{W}_{ex_i} \right\|^2 \right] \\
&+ \sum_{i=1}^{N_i} \left[\frac{1}{2}\beta(i) \left\| \begin{array}{c} W_i^* \\ W_{ex_i}^* \end{array} \right\|^2 + \frac{1}{2}\epsilon^2(i) + \iota \varpi_i \right]
\end{aligned} \tag{59}$$

where, $K_3 = K_2 - \frac{1}{2}$. By combining Equations (40), (46) and (54), we can get:

$$V = Q + \frac{1}{2} z_2^T M(\theta) z_2 + \frac{1}{2}\sum_{i=1}^{N_i} \widetilde{W}_i^T \Lambda^{-1} \widetilde{W}_i \tag{60}$$

where, $Q = \sum_{i=1}^{N_d} \left[\ln \frac{1}{(1-\chi_i^2)} \right]$. Therefore, inequality (59) can be re-expressed as:

$$\dot{V}(t) = \eta_s V(t) + \mu_s \tag{61}$$

where, $\eta_s = \min\left\{ \lambda_{\min}(K_1), \frac{2\lambda \min(K_3)}{\lambda(M(\theta))}, \frac{\beta}{\lambda_{\max}(\Lambda^{-1})} \right\}$, $\mu_S = \sum_{i=1}^{N_i} \left(\frac{1}{2} \left\| W_i^* \right\|^2 + \frac{1}{2}\epsilon^2(i) + \iota \varpi_i \right)$. We make $S_s = \frac{\mu_s}{\eta_s}$. By combining equations (60) and (61), we can get:

$$\begin{aligned}
\dot{V}(t) &\leq (V(0) - \varsigma_s) \exp(-\eta_s t) + \varsigma_s \\
&\leq V(0) + \varsigma_s
\end{aligned} \tag{62}$$

So $\dot{V}(t)$ is negative definite, and the joint velocity tracking error z_2 and the neural network weight estimation error $\widehat{W}$ are bounded. Thus it can be deduced that the position tracking error z_1 is also bounded and meets the preset tracking performance, the system states ξ and the Jacobian $J(\theta)$ are also bounded. Therefore, we can obtain that all state signals of the robot system can meet the final consistency stability no matter in the training stage or in the neural network expansion stage. Therefore, the system is stable in the Lyapunov sense [31].

4. Experiment results

In experiment 1, the effectiveness of the force controller was demonstrated by a tabletop wipe experiment. The robot first wiped the desktop from right to left, and then from left to right. We conducted experiments with and without force controller

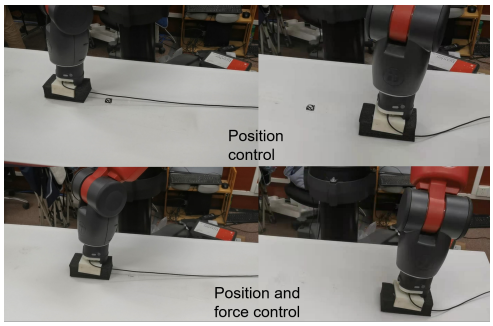


Figure 2: The experimental results of the comparative test show that the desired task cannot be achieved only by using position control, and the task can be successfully completed by introducing contact force.

respectively, and the experimental results are shown in Fig. 2. It can be found that the handwriting on the desktop cannot be wiped clean when only the position control is used, but the handwriting on the desktop can be cleaned smoothly when the force controller is added. Figure 2 shows the force tracking results. Here we give the expected force as a constant force of

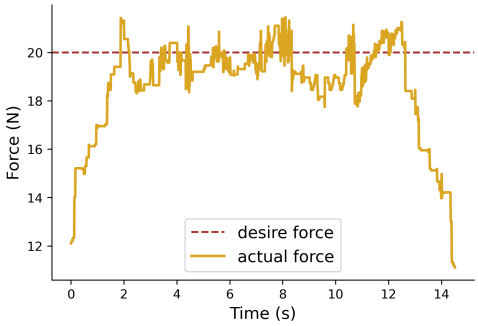
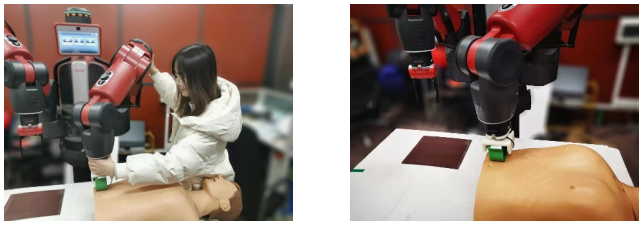


Figure 3: The force tracking of the manipulator when it performs the task after the force controller is added.

20N. It can be seen that the force error during the whole experiment can be maintained within 2N.

In experiment 2, Baxter robot autonomous ultrasound scanning task was used to prove the effectiveness of the proposed scheme. The whole experiment process is divided into demonstration stage, reproduction stage and generalization stage.

In the demonstration stage, the instructor dragged the manipulator arm to perform body ultrasound task, as shown in Fig.4(a). The operation was repeated four times and the force and trajectory of the end-effector were recorded. In the model



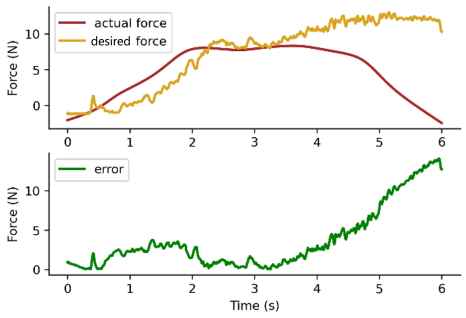
(a) (b)

Figure 4: (a) and (b) are the demonstration process and repetition process respectively.

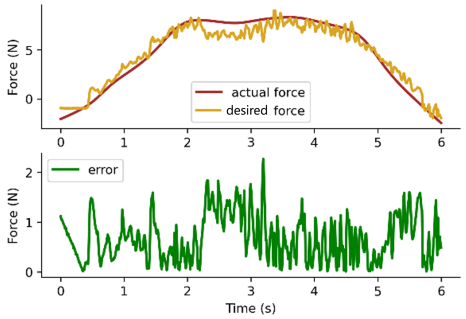
learning stage, GMR is used to fit the motion trajectories of X and Y axes as well as the force trajectories of the Z-axis, and a trajectory containing more motion information is obtained, as shown in Fig. 5. Then we used DMPs to learn the motion and force trajectories after processed by GMM and GMR to get the DMPs model parameters.

In the reproduction stage, the robot performs ultrasound scanning task according to the demonstration, as shown in Fig. 4 (b). In order to prove the effectiveness of the proposed force controller, we compare the force tracking trajectory without the addition of force controller and with the addition of a force control. As shown in Fig. 6, we can find that the error between the actual force and the expected force exceeds 10N when the force controller is not added. In the process of ultrasound scanning, excessive force will make patients feel uncomfortable. When the force controller is added, the error between the expected force and the actual force can be controlled within 2N, which makes the robot more stable when performing ultrasound task.

In addition, the broad neural network controller is used to control the movement of the manipulator, and the force tracking in the process of reproduction is shown in Fig. 7.



(a)



(b)

Figure 6: (a) represents the force tracking error in the absence of a force controller; (b) shows the force tracking error after the addition of force controller.

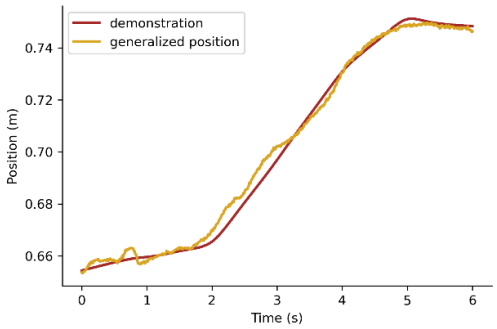


Figure 7: Position tracking under the control of broad neural network in the reproduction stage.

In the generalization stage, we generalized the motion track and force track of the teaching respectively, to adapt to the ultrasound scanning of different positions and different forces. We used DMPs to generalize the learned position trajectory of the x-axis and force trajectory of the z-axis, as shown in Fig. 8.

The position tracking trajectory in the generalization process is shown in the Fig. 9. In the generalization process, we no longer retrain the parameters of the broad neural network, but use the parameters learned in the reproduction process. Since

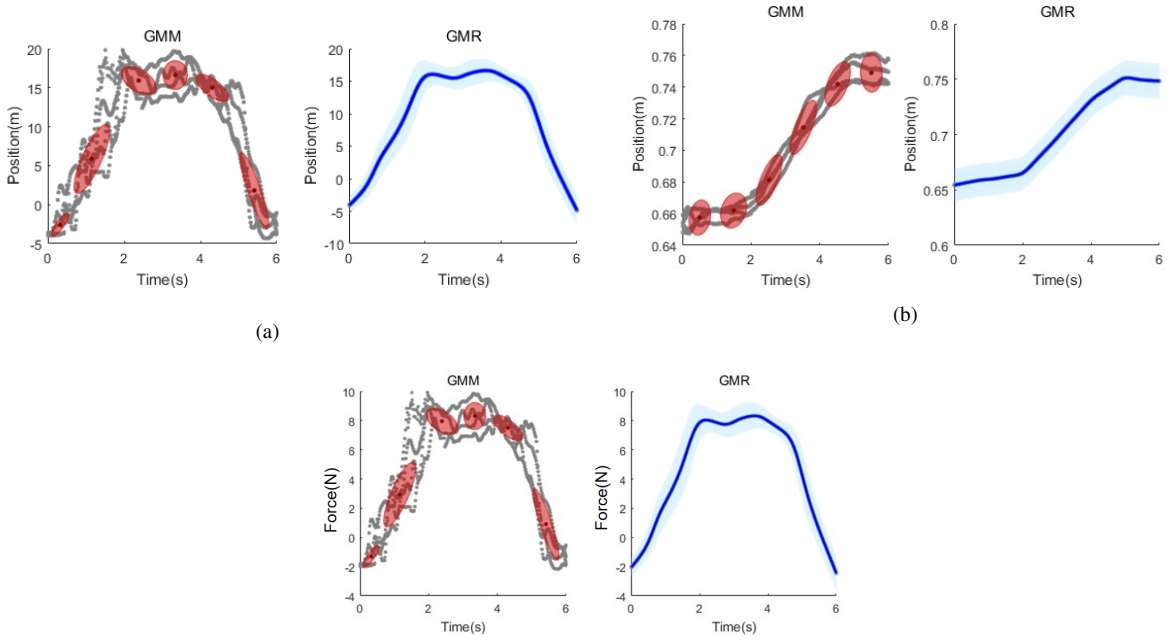


Figure 5: GMM and GMR were used to fit the teaching trajectories. (a) (b) are respectively the motion trajectory of the end-effector of the manipulator on the X, Y axis component, (c) is the force trajectory of the object. Trajectories (blue lines) are generated based on demonstration trajectories (gray dots).

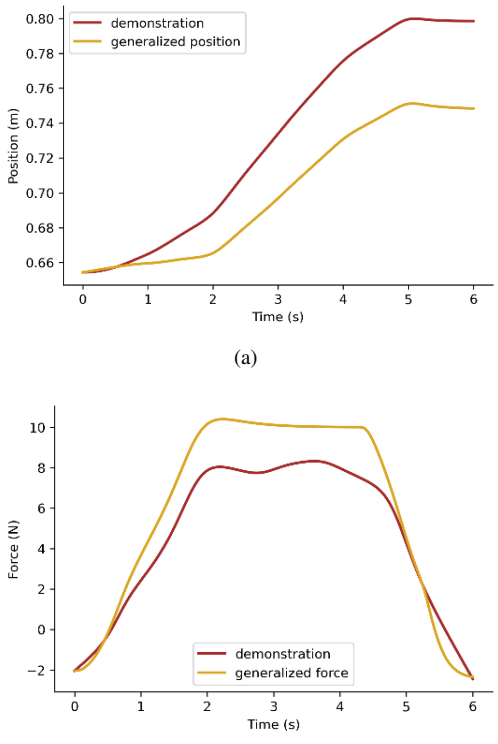
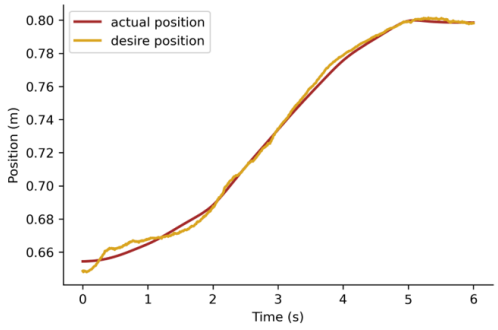


Figure 8: (a) and (b) are the generalization of motion trajectory and force trajectory respectively.

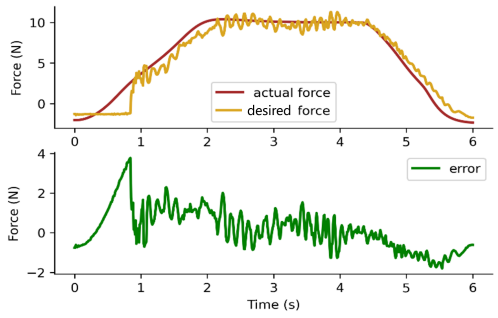
the robot's trajectory in the generalization process exceeds the compact set of the neural network, the broad neural network will track the expected trajectory by adding nodes when performing the generalization task. It can be seen from Fig. 9 (a) that the trajectory other than the learning trajectory can still be well tracked. Fig. 9 (b) shows the force tracking during the execution of the generalized task. It can be seen from Fig. 9 (b) that the error between the actual force and the expected force can still be controlled within 2N during the execution of the generalized task.

5. Conclusion

Traditional neural networks such as RBFNN neural network cannot achieve the ideal approximation effect when the input exceeds the compact set of the neural network. In this paper, the broad neural network is used to replace the traditional neural network. When the input exceeds the compact set of the neural network, the broad neural network will expand the compact set by adding nodes. We use a broad neural network to track the desired trajectory generated by DMPs, which can ensure that the trajectory of DMPs generalization can still be accurately tracked. On the basis of position control, we propose a robot skill learning framework with force control, which is used to track the force in the process of reproduction and generalization. Finally, the method is applied to the ultrasound task, and the effectiveness of the proposed method is proved by experiments.



(a)



(b)

Figure 9: (a) shows the position tracking under the control of broad neural network in the generalization stage; (b) represents the force tracking in the generalization phase

Acknowledgement

This work was partially supported by Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/S001913 and the Natural Science Foundation of Jiangsu Province (Grant No. BK20180235).

References

[1] A. J. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), Vol. 2, IEEE, 2002, pp. 1398–1403.

[2] S. Schaal, Dynamic movement primitives-a framework for motor control in humans and humanoid robotics, in: Adaptive motion of animals and machines, Springer, 2006, pp. 261–280.

[3] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, D. G. Caldwell, Statistical dynamical systems for skills acquisition in humanoids, in: 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), IEEE, 2012, pp. 323–329.

[4] S. Schaal, A. Ijspeert, A. Billard, Computational approaches to motor learning by imitation, Philosophical Transactions of the Royal Society of London 358 (1431) (2003) 537–47.

[5] E. Theodorou, J. Buchli, S. Schaal, A generalized path integral control approach to reinforcement learning, Journal of Machine Learning Research 11 (2010) 3137–3181.

[6] Ruohan Wang, Y. Wu, Wei Liang Chan, Keng Peng Tee, Dynamic movement primitives plus: For enhanced reproduction quality and efficient trajectory modification using truncated kernels and local biases, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 3765–3771. doi:10.1109/IROS.2016.7759554.

[7] T. Cederborg, M. Li, A. Baranes, P.-Y. Oudeyer, Incremental local online gaussian mixture regression for imitation learning of multiple tasks, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 267–274.

[8] S. Calinon, T. Alizadeh, D. G. Caldwell, On improving the extrapolation capability of task-parameterized movement models, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 610–616.

[9] T. Alizadeh, S. Calinon, D. G. Caldwell, Learning from demonstrations with partially observable task parameters, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 3309–3314.

[10] Y. Liu, X. Chen, Y. Mei, Y. Wu, Observer-based boundary control for an asymmetric output-constrained flexible robotic manipulator, SCIENCE CHINA Information Sciences.

[11] H. Huang, C. Yang, C. L. P. Chen, Optimal robot-environment interaction under broad fuzzy neural adaptive control, IEEE Transactions on Cybernetics PP (99) (2020) 1–12.

[12] J. Na, J. Zhao, G. Gao, Z. Li, Output-feedback robust control of uncertain systems via online data-driven learning, IEEE Transactions on Neural Networks and Learning Systems PP (99) 1–13.

[13] Y. Liu, F. Guo, X. He, Q. Hui, Boundary control for an axially moving system with input restriction based on disturbance observers, IEEE Transactions on Systems, Man, and Cybernetics: Systems 49 (11) (2019) 2242–2253. doi:10.1109/TSMC.2018.2843523.

[14] H. Mohammadi, H. Shiri, Adaptive optimal tracking control for a class of nonlinear systems with fully unknown parameters, in: 2017 5th International Conference on Control, Instrumentation, and Automation (ICCIA), IEEE, 2017, pp. 337–342.

[15] C. Yang, Y. Jiang, Z. Li, W. He, C. Y. Su, Neural control of bimanual robots with guaranteed global stability and motion precision, IEEE Transactions on Industrial Informatics 13 (3) (2017) 1162–1171.

[16] H. Huang, T. Zhang, C. Yang, C. L. P. Chen, Motor learning and generalization using broad learning adaptive neural control, IEEE Transactions on Industrial Electronics 67 (10) (2020) 8608–8617.

[17] H. Huang, T. Zhang, C. Yang, C. L. P. Chen, Motor learning and generalization using broad learning adaptive neural control, IEEE Transactions on Industrial Electronics 67 (10) (2020) 8608–8617.

[18] G. Peng, C. Yang, W. He, C. P. Chen, Force sensorless admittance control with neural learning for robots with actuator saturation, IEEE Transactions on Industrial Electronics 67 (4) (2019) 3138–3148.

[19] G. Ganesh, A. Albu-Schäffer, M. Haruno, M. Kawato, E. Burdet, Biomimetic motor behavior for simultaneous adaptation of force, impedance and trajectory in interaction tasks, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 2705–2711.

[20] E. Gribovskaya, A. Kheddar, A. Billard, Motion learning and adaptive impedance for robot control during physical interaction with humans, in: 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 4326–4332.

[21] C. Yang, C. Chen, W. He, R. Cui, Z. Li, Robot learning system based on adaptive neural control and dynamic movement primitives, IEEE Transactions on Neural Networks and Learning Systems 30 (2019) 777–787.

[22] S. S. Ge, T. H. Lee, C. J. Harris, Adaptive Neural Network Control of Robotic Manipulators, WORLD SCIENTIFIC, 1998.

[23] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the Royal Statistical Society: Series B (Methodological) 39 (1) (1977) 1–22.

[24] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, D. G. Caldwell, Statistical dynamical systems for skills acquisition in humanoids, in: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), 2012, pp. 323–329.

[25] M. Y. Wu, N. M. Arbouz, Control of robot manipulators, in: Circuits, Systems and Computers, 1985. Nineteeth Asilomar Conference on, 2002.

[26] J. Park, I. Sandberg, Universal approximation using radial-basis-function networks, Neural Computation 3 (2) (2014) 246–257.

[27] J. Patra, R. Pal, B. Chatterji, G. Panda, Identification of nonlinear dynamic systems using functional link artificial neural networks, IEEE Trans Syst Man Cybern B Cybern 29 (2) (1999) 254–262.

[28] C. Yang, Y. Jiang, Z. Li, W. He, C.-Y. Su, Neural control of bimanual robots with guaranteed global stability and motion precision, IEEE Transactions on Industrial Informatics 13 (3) (2016) 1162–1171.

[29] J. Na, J. Zhao, G. Gao, Z. Li, Output-feedback robust control of uncer-

tain systems via online data-driven learning, IEEE Transactions on Neural Networks and Learning Systems PP (99) 1–13.

[30] Y. Liu, Y. Fu, W. He, Q. Hui, Modeling and observer-based vibration control of a flexible spacecraft with external disturbances, IEEE Transactions on Industrial Electronics 66 (11) (2019) 8648–8658. doi:10.1109/TIE.2018.2884172.

[31] Y. Liu, W. Zhan, M. Xing, Y. Wu, R. Xu, X. Wu, Boundary control of a rotating and length-varying flexible robotic manipulator system, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2020) 1–10doi:10.1109/TSMC.2020.2999485.