Neurocomputing 521 (2023) 146–159

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A trajectory and force dual-incremental robot skill learning and generalization framework using improved dynamical movement primitives and adaptive neural network control

Zhenyu Lu [a], Ning Wang [a], Qinchuan Li [b], Chenguang Yang [a,*]

[a] Bristol Robotics Laboratory, University of the West of England, Bristol BS16 1QY, United Kingdom
[b] School of Mechanical Engineering, Zhejiang Sci-Tech University, Hangzhou, Zhejiang Province, 310018, China

ARTICLE INFO

Article history:
Received 10 June 2022
Revised 26 October 2022
Accepted 21 November 2022
Available online 2 December 2022
Communicated by Zidong Wang

Keywords:
Incremental skill learning and generalization
Learning from demonstration
Dynamic movement primitive (DMP)
Adaptive neural network (NN) control
Multiple stylistic skill generalization

ABSTRACT

Due to changes in the environment and errors that occurred during skill initialization, the robot's operational skills should be modified to adapt to new tasks. As such, skills learned by the methods with fixed features, such as the classical Dynamical Movement Primitive (DMP), are difficult to use when the using cases are significantly different from the demonstrations. In this work, we propose an incremental robot skill learning and generalization framework including an incremental DMP (IDMP) for robot trajectory learning and an adaptive neural network (NN) control method, which are incrementally updated to enable robots to adapt to new cases. IDMP uses multi-mapping feature vectors to rebuild the forcing function of DMP, which are extended based on the original feature vector. In order to maintain the original skills and represent skill changes in a new task, the new feature vector consists of three parts with different usages. Therefore, the trajectories are gradually changed by expanding the feature and weight vectors, and all transition states are also easily recovered. Then, an adaptive NN controller with performance constraints is proposed to compensate dynamics errors and changed trajectories after using the IDMP. The new controller is also incrementally updated and can accumulate and reuse the learned knowledge to improve the learning efficiency. Compared with other methods, the proposed framework achieves higher tracking accuracy, realizes incremental skill learning and modification, achieves multiple stylistic skills, and is used for obstacle avoidance with different heights, which are verified in three comparative experiments.

1. Introduction

In recent decades, learning from demonstrations (LfD), a technique that develops strategies from example states to action mappings [1], has attracted considerable attention along with the development of robotics and AI technologies. A recent survey on LfD concluded that current limitations of LfD include representation of complex behaviours, reliance on labelled data, and suboptimal and inappropriate demonstrators [2]. To solve this problem, this paper proposes an incremental skill learning and generalization framework to enable robots to modify simple initial actions to complex cases. This method is based on motion primitive (MP) technology, in which a long-term complex motion is divided into multiple sub-actions. Then, the sub-actions are extracted into MPs and finally these MPs are reprogrammed and generalised to fit a new task [2].

The MP can be presented in many forms, e.g. Kernelized Movement Primitive (KMP) [3], Compliant Movement Primitive (CMP) [4] and Dynamical Movement Primitive (DMP) [5]. DMP was proposed by Ijspeert et al [6,7] and then improved by many researchers. In addition to the classical DMP, there are a number of improved methods such as discrete DMP, periodic DMP, etc., and some scholars combined DMP with reinforcement learning (RL) [8,9], deep learning [10], life-long learning and various control methods [11–13] to expand the scope of DMP. DMP has a very concise expression that is a second-order function with only three variables and a forcing function. And the applications of DMP contain trajectory tracking in Euclidean space [14], EMGs signal prediction [21], force control in a contact manipulation [22], motion and state monitoring [23] and special tasks such as obstacle avoidance [15,17], cooperative manipulations [16,24] and multi-modal skill learning.

* Corresponding author.
E-mail address: cyang@ieee.org (C. Yang).

https://doi.org/10.1016/j.neucom.2022.11.076
0925-2312/

The limitation of the classical DMP is that once the skills are learned, the characteristics expressed by the forcing function are fixed. Even though some variables, e.g. position, velocity, can be generalized in space and time by modifying the starts, goals and scaling factors. Some improvements of DMP in [15–17,24] are made by adding additional terms for obstacle avoidance and cooperative manipulation. However, the terms are specially designed by using time-related variables such as position, angle, and velocity, etc., which cannot be generalized in phase space, like 's' in the forcing function. If operational requirements keep changing, the newly learned skills and added terms should update, which costs a lot of time and increases the complexity of computation. Reinforcement learning is used to achieve DMP-based incremental skill learning. For example, Matteo et al. proposed an incremental point-to-point motions learning method based on a dynamical system. For a new demonstration, the original dynamical system will be redesigned to approach the new task [25]. Lemme et al. proposed a bootstrapping cycle to build a suitable primitive library [26]. In this library, the old primitives are refined and new ones are added, while the unused ones are deleted. Yuan et al. [8] and Li et al. [9] followed the similar technique and updated weights by integrating probability-weighted RL to realize skill modification. Wang and Wu et al. [27,29] proposed DMP plus (DMP+) method to realize efficient skill modifications by using truncated kernels and local biases to achieve two contributions. One is preserving the desirable properties of the original skill and achieving lower mean square errors (MSEs). The other is the reusability of existing primitives, which can reduce human fatigue in imitation learning and correcting errors in demonstration without requiring further demonstration. Compared with RL-based methods, DMP+ requires less computation and retains the original features, which is used as a benchmark method for comparison in this method.

The combination of DMP and robot control is another topic that attracts much attention. Schaal et al., [6,31], proposed a framework for motor control combining DMP. In our previous research, we combined DMP and adaptive NN control [21], admittance control [32], and neural networks [35,36] for robot control. In this paper, inspired by DMP+, we propose a novel incremental skill learning and generalization method called incremental DMP (IDMP). The forcing function of IDMP can be incrementally updated by adding new features and weights to track new trajectories. Considering uncertain dynamics parameters and state tracking limitations, an adaptive NN controller is designed, in which the NN term is also incrementally updated and can accumulate and reuse the learned knowledge to improve the learning efficiency. System stability are ensured by building a barrier Lyapunov function (BLF). The proposed framework is shown in Fig. 1: First, an old trajectory is expressed by DMP with a forcing function$f(s)$. After linear transformation of the feature vector, we can obtain the incremental term λ_k to compose a IDMP function to achieve a new trajectoryx^N, which is then transferred by an inverse kinematic solver to obtain the joint information as the input of the control part. The real-time joint tracking errors are applied to create a virtual controller by BLF. We consider dynamics uncertainties and contact force estimation errors, and use adaptive control term to estimate and compensate the errors based on incremental adaptive NN control to ensure system stability.

Compared to DMP-based trajectory planning and various control methods, the proposed framework offers three advantages:

a) Skill incremental learning and original skill preservation
Similar to DMP + and Acnmp [38], the old skills can be preserved during the gradual adaptation process to new situations, so they can be easily recovered for the old situations. The difference in the computation from DMP + is the skill adaptation is realized by adding new linear transformations of the existing kernels rather than changing kernels, which gives the forcing function with a stronger nonlinear adaptability and makes it more suitable for the dynamic skill learning process without adding new kernels.
b) High-accuracy trajectory tracking and multi-style skill transformation
According to the board learning in [19], the preliminary NN can achieve better performance after inserting additional extension nodes, which have a similar function as the linear transformations of IDMP. Therefore, we use IDMP to improve trajectory tracking performance and realize multi-style skill transformation. Multi-style skill transformation suits the situation that the original skill is ambiguous and leads to different styles in motion sequence [37]. An example, like the following second experiment, is a letter recognized as an 'a' first, and it is probably a 'u' or a 'v' after confirmation. We show the skill transformation process from one shape to multi-stylistic shapes based on the same original and extended features, but with different weights.
c) Adaptive NN control with constraints on the transient tracking errors
After renewing trajectory using IDMP, robot system controller should be improved to minimize tracking errors to the updated trajectory. Additionally, the controller should process the uncertain dynamics parameters, force estimation errors, and limitations on the transient state errors. In this paper, we proposed an adaptive NN and BLF-based controller, where the NN nonlinear fitting part can increase the

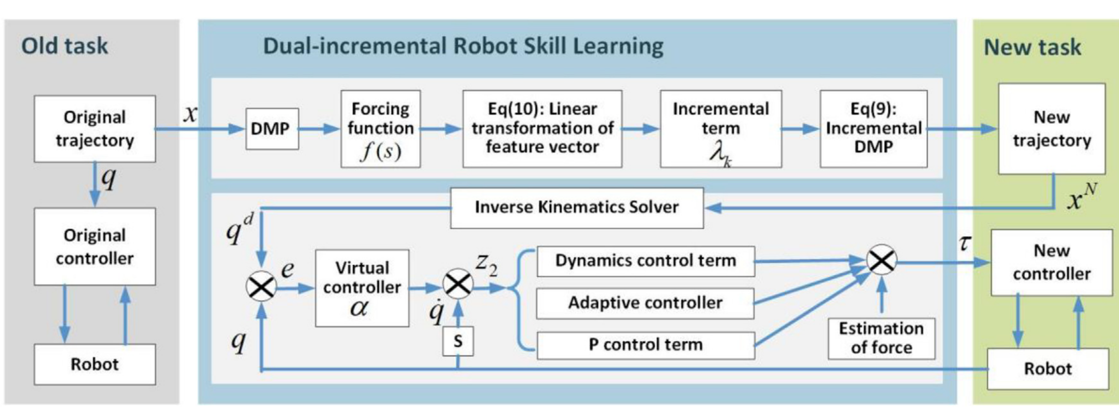


Fig. 1. Diagram of trajectory and force dual-incremental robot skill learning and generalization.

number of neurons and update the weights, so that it can accumulate and reuse the learned weights to improve the learning efficiency.

The remainder of the paper is organized as follows: Section II briefly introduces DMP. In Section III, we present the details of IDMP and extend it to the multi-style skill learning. In Section IV, we propose the new adaptive NN controller with constraints on transient tracking errors. In Section V, three experiments are conducted to verify the above advantages. Section VI provides a final conclusion.

2. Related work to dynamical movement primitive

The DMP model proposed by Ijspeert et al., [6,7] is

$$\begin{cases} \tau \dot{v} = K(g-x) - Dv + (g-x_0)f(s) \\ \tau \dot{x} = v \end{cases}, \tag{1}$$

where $K, D > 0$ are stiffness and damping factors and $\tau > 0$ is a timing parameter for adjusting duration of the trajectory, x_0 and g are start and end of the trajectory. $f(s) = \theta^T \Psi(s)$ is a linear combination of the normalized Gaussian functionsψ_i, where$\theta = [w_1, w_2, \ldots, w_n]^T$, $\Psi(s) = [\psi_1, \psi_2, \ldots, \psi_n]^T$, and w_i is the weight of ψ_i and the Gaussian functions ψ_i is expressed as

$$\psi_i = \frac{\varphi_i(s)s}{\sum_{i=1}^{n} \varphi_i(s)}, \ \varphi_i(s) = \exp\left(-h_i(s-c_i)^2\right), \tag{2}$$

where c_i and $h_i > 0$ are the centre and width of the radial basis function$\varphi_i(s)$. The transformation function (or forcing function)$f(s)$ is expressed by the phase variable s and a canonical system

$$\tau \dot{s} = -os, \quad o > 0. \tag{3}$$

The converging time is modified by factor o to ensure $s \to 0$ at the end state for erasing the influence of $f(s)$ in. The θ is (1) estimated by minimizing the function

$$\min\left(\sum_{k=1}^{N}\left(f_k^{Tar} - f(s)\right)^2\right) \tag{4}$$

where $f^{Tar}(s)$ the target value of $f(s)$ that is calculated by the kth, $k = 1, 2, \ldots, N$ demonstrated trajectory x_d^k and velocityv_d^k:

$$f_k^{Tar} = (\tau \dot{v}_d - K(g - x_d^k) - Dv_d^k)/(g - x_0) \tag{5}$$

Remark 1. *DMP method is also applied multi-style skill learning from multi-demonstrations, named Stylistic DMP (SDMP) [37]. The SDMP modifies (1) into.*

$$\begin{cases} \tau \dot{v} = K(g-x) - Dv + (g-x_0)\tilde{f}(s) \\ \tau \dot{x} = v \end{cases} \tag{6}$$

where $\tilde{f}(s)$ is then expressed as$\tilde{f}(\mathbf{s}) = \sum_{j=1}^{J} f_j(s_j) f_j(s_j) = \theta_j^T \Psi_j(s_j)$, $\theta_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^T$, $\Psi_j(s_j) = [\psi_{j1}, \psi_{j2}, \ldots, \psi_{jn}]^T$,$\psi_{ji} = \frac{\varphi_{ji}(s_j)s_j}{\sum_{i=1}^{n} \varphi_{ji}(s_j)}$, $\varphi_{ji}(s_j) = \exp(-h_{ji}(s_j - c_{ji})^2)$. Set $\mathbf{s} = [s_1, s_2, \ldots, s_J]$ as a style parameter vector and $\Theta = [\theta_1, \theta_2, \ldots, \theta_J]$ as a parameter matrix, the calculation purpose is to acquire the optimal parameter vector$\Theta* = [\theta_1*, \theta_2*, \ldots, \theta_J*]$.

Considering different skills expressed by DMP have the same expression as in (1), and the main difference focuses on the forcing function$f(s)$, we can update the $f(s)$ to realize skill incremental learning. Following the idea of board learning system (BLS) [19,20], an efficient incremental learning system without the need for deep architecture, incremental learning algorithm has a promising performance in calculation accuracy and learning speed. Especially, with the increase of the enhancement nodes, the network can approach a nonlinear function with any accuracy, which inspires us to build an incremental updating forcing function that the vectors θ and $\Psi(s)$ can be extended to change the learned skills and fit new trajectories.

The main challenge is how to reshape θ and $\Psi(s)$ to enable the new added features and extended terms to satisfy the properties of DMP. For example, in (2), the feature variables $\psi_i, i = 1, 2, \ldots, m$ are normalized and satisfy

$$\begin{aligned} \sum_{i=1}^{n} \psi_i &= \frac{\varphi_1(s)s}{\sum_{i=1}^{n} \varphi_i(s)} + \frac{\varphi_2(s)s}{\sum_{i=1}^{n} \varphi_i(s)} + \ldots + \frac{\varphi_n(s)s}{\sum_{i=1}^{n} \varphi_i(s)} \\ &= \frac{\sum_{i=1}^{n} \varphi_i(s)s}{\sum_{i=1}^{n} \varphi_i(s)} \\ &= s \end{aligned} \tag{7}$$

If we set an extended function as $\Phi_j, j = 1, 2, \ldots, m$ and $\hat{\psi}_i$ as the modified term to$\psi_i, i = 1, 2, \ldots, n$, they will be normalized and satisfy:

$$\sum_{i=1}^{n} \hat{\psi}_i + \sum_{j=1}^{m} \Phi_j = s \tag{8}$$

Then the main question in IDMP is how to generate Φ_j and modify $\hat{\psi}_i$ to enable (8) is satisfied. A lemma is presented for the following deduction process.

Lemma 1. *For matrices$A \in R^{n \times m}$,$W \in R^{m \times 1}$ and $Y \in R^{n \times 1}$ satisfying$Y = AW$, if A is extended to$\bar{A} = [A|a] \in R^{n \times (m+k)}$, and$a \in R^{n \times k}$, then the new weight vector $\bar{W} \in R^{(m+k) \times 1}$ is calculated based on the W as.*

$$\bar{W} = \begin{bmatrix} W - db^T Y \\ b^T Y \end{bmatrix} \tag{9}$$

where $d = (A)^+ a$,$b^T = \begin{cases} c^+ & \text{if } c \neq 0 \\ (1 + d^T d)^{-1} d^T (A)^+ & \text{if } c = 0 \end{cases}$, $c = a - Ad$.

3. Incremental dynamical movement primitive

3.1. Basic incremental dynamical movement primitive

Similar to (1), we define a new skill expressed by DMP in (10) that is different from the skills learned from the original demonstration. Given new position x^N and velocityv^N, the new skill is expressed as

$$\begin{cases} \tau \dot{v}^N = K(g - x^N) - Dv^N + (g - x_0)f^N(s) \\ \tau \dot{x}^N = v^N \end{cases}, \tag{10}$$

where $f^N(s) = (\theta^N)^T \Psi^N(s)$ is a new nonlinear function to be learned, andτ, K and D are as the same as those in (1).

For a new trajectory, the previous method will redefine a new pair of vectors θ^N and $\Psi^N(s)$ or add new kernels to DMP to adapt to novel situations [29]. DMP + smartly reused and modified the kernels and weights for skill efficient adaptation to avoid re-calculation [27,29]. In IDMP, we create extended feature terms $\eta_j(s)$ by making a linear transformation to the feature vector$\Phi(s) = [\varphi_1(s), \varphi_2(s), \ldots, \varphi_n(s)]$:

$$\eta_j(s) \equiv \zeta_j(\Phi(s)W_{ej} + \beta_{ej}), \quad j = 1, 2, \ldots, m, \tag{11}$$

where ζ_j represents the jth transformation function of $\Phi(s)$, and W_{ej} and β_{ej} are random weights and bias terms. Then the new feature terms $\eta_j(s)$ are used in combination with $\hat{\psi}$ to achieve λ_k (12) in to contribute to the output $f^N(s)$ (see in Fig. 2).

It is obvious that $\sum_{j=1}^m \eta_j(s) + \sum_{i=1}^n \psi_i \neq s$, such that the property (8) is not satisfied, but the property (8) is the insurance that the final state value x converging monotonically to g. Therefore, using the new term $\eta_j(s)$ and ψ_i, we build a new term λ_k as

$$\lambda_k = \frac{q_k(s)\varphi_i(s) + (1-q_k(s))\eta_j(s)}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)} s, \quad k = 1, \ldots, m+n, \tag{12}$$

where $q_k(s) = 1$, if $k \in [1, n]$ and $q_k(s) = 0$, if $k \in [n+1, n+m]$.

It is obvious $\sum_{k=1}^{m+n} \lambda_k = s$, and (12) if $k \in [n+1, n+m]$, can be simplified as

$$\lambda_k = \frac{\eta_j(s)}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)} s, \tag{13}$$

which is a modified term of $\eta_j(s)$ and equals to the Φ_j in (8). If $k \in [1, n]$, (12) can be simplified as

$$\begin{aligned} \lambda_k &= \frac{\varphi_i(s)}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)} s, \\ &= \frac{\varphi_i(s)s}{\sum_{i=1}^n \varphi_i(s)} \cdot \frac{\sum_{i=1}^n \varphi_i(s)}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)} \\ &= \frac{\sum_{i=1}^n \varphi_i(s)}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)} \psi_k \end{aligned} \tag{14}$$

which means the term ψ_k is scaled up $\frac{\sum_{i=1}^n \varphi_i(s)}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)}$ times and can be seen as the $\hat{\psi}_k$ in (8) to represent the modified ψ_k. Furthermore, we set a new scaling variable Γ as

$$\Gamma = \frac{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)}{\sum_{i=1}^n \varphi_i(s)} \tag{15}$$

Then, we can get $\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s) = \frac{\Gamma}{\Gamma-1} \sum_{j=1}^m \eta_j(s)$, and λ_k in (12) is rewritten as

$$\begin{aligned} \lambda_k(s) &= \frac{q_k(s)\varphi_i(s)s}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)} + \frac{(1-q_k(s))\eta_j(s)s}{\sum_{i=1}^n \varphi_i(s) + \sum_{j=1}^m \eta_j(s)} \\ &= \frac{q_k(s)\varphi_i(s)s}{\Gamma \sum_{i=1}^n \varphi_i(s)} + \frac{(\Gamma-1)(1-q_k(s))\eta_j(s)s}{\Gamma \sum_{j=1}^m \eta_j(s)} \\ &= \frac{q_k(s)\psi_i(s)}{\Gamma} + \frac{(\Gamma-1)(1-q_k(s))\eta_j(s)s}{\Gamma \sum_{j=1}^m \eta_j(s)} \end{aligned} \tag{16}$$

Here, we set

$$\gamma_j(s) = \frac{\eta_j(s)s}{\sum_{j=1}^m \eta_j(s)}, \tag{17}$$

Then λ_k in (16) is further expressed as

$$\begin{aligned} \lambda_k(s) &= \underbrace{q_k(s)\frac{\psi_i(s)}{\Gamma}}_{\text{modified original skill}} + \underbrace{(1-q_k(s))\frac{(\Gamma-1)\gamma_j(s)s}{\Gamma}}_{\text{incremental skill}} \\ &= \underbrace{q_k(s)\psi_i(s)}_{\text{original skill}} + \underbrace{q_k(s)\frac{(1-\Gamma)\psi_i(s)}{\Gamma}}_{\text{old skill modification}} + \underbrace{(1-q_k(s))\frac{(\Gamma-1)\gamma_j(s)s}{\Gamma}}_{\text{incremental skill}} \end{aligned} \tag{18}$$

Remark 2. *Different from trajectory modification methods by adding extra terms, e.g.* $\gamma RV\varphi \exp(-\beta\varphi)$ *in [15] (V, φ are physical variables to represent velocity and joint), to the DMP function,* λ_k *and* $f^N(s)$ *are updated by adding new feature terms* $\eta_j(s)$*, which have nonlinear relationship with* ψ_i *in (2). The process is somewhat similar to the truncating kernels in DMP +. While the difference is, after adding* $\eta_j(s)$*,* $f^N(s)$ *can keep updating and the learned skills can approach the desired trajectory with any accuracy, which can be explained by the principles of incremental learning in [20]. However, DMP + depends on but is constrained by the limited kernels. The compare of two methods will be further performed through the following experiment.*

Seen from (18), if $m = 0$, we have $\lambda_i(s) = \psi_i(s)$, which is a standard DMP term. After adding more new terms of $\gamma_j(s)$, the effect of $\psi_i(s)$ changes and the number of $\lambda_k(s)$ increases. Here, we set θ^N and $\Psi^N(s)$ in (10) as $\theta^N = [w_1, w_2, \ldots, w_n, w_{n+1}, \ldots, w_{n+m}]^T$ and $\Psi(s) = [\lambda_1, \lambda_2, \ldots, \lambda_{m+n}]^T$. Each λ_k consists of three parts that are

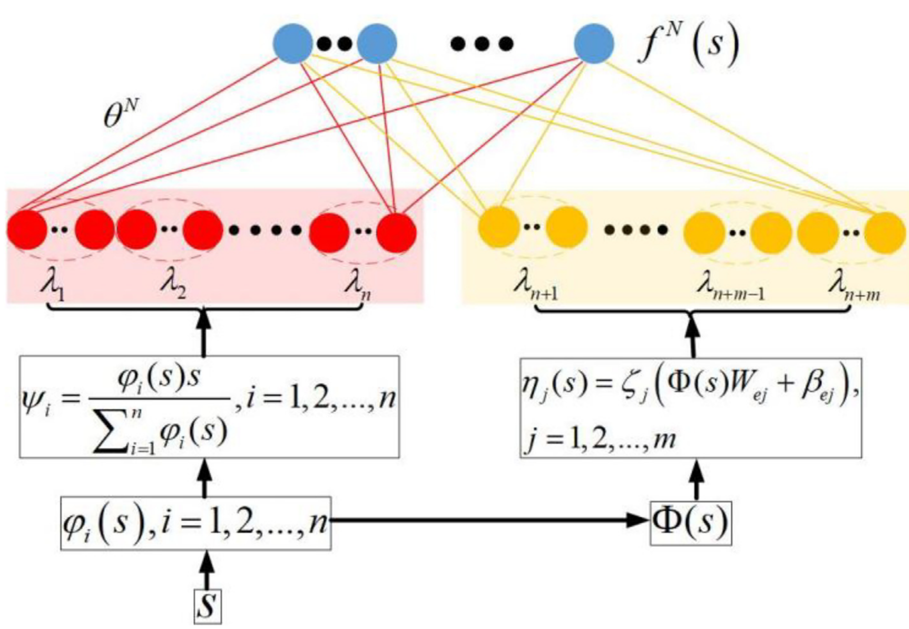


Fig. 2. Diagram of incremental dynamical movement primitive.

marked as the original skill$q_k(s)\psi_i(s)$, the old skill modification $q_k(s)\psi_i(s)$ with a coefficient $(1-\Gamma)/\Gamma$ and incremental skill generated by the normalized Gaussian function$\gamma_j(s)$. Setting$\Upsilon(s)=[\gamma_1(s),...,\gamma_m(s)]$, then the new forcing function$f^N(s)$ is:

$$\begin{aligned} f^N(s) &= \sum_{k=1}^{m+n}(w_k)^T\lambda_k(s) \\ &= \left(\theta^N\right)^T\Psi^N(s) \\ &= \underbrace{\theta^T\Psi(s)}_{\text{original skill}} + \underbrace{(1/\Gamma-1)\left(\theta^C\right)^T\Psi(s)}_{\text{old skill modification}} + \underbrace{(1-1/\Gamma)\left(\theta^U\right)^T\Upsilon(s)}_{\text{enhancement skill}}, \end{aligned} \tag{19}$$

where θ^N and $\Psi^N(s)$ are new weight and state vectors, which are expressed as

$$\begin{aligned} \theta^N &= \left[w_1,...,w_n|w_1^c,...,w_n^c|u_1,...,u_m\right]^T \\ &= \left[\theta|\theta^C|\theta^U\right]^T \end{aligned} \tag{20}$$

$$\begin{aligned} \Psi^N(s) = [&\psi_1(s),...,\psi_n(s)|(1/\Gamma-1)\psi_1(s),...,(1/\Gamma-1)\psi_n(s) \\ &|(1-1/\Gamma)\gamma_1(s),...,(1-1/\Gamma)\gamma_m(s)]^T \\ = [&\Psi(s)|(1/\Gamma-1)\Psi(s)|(1-1/\Gamma)\Upsilon(s)]^T \end{aligned} \tag{21}$$

where $\theta^C=[w_1^c,w_2^c,...,w_n^c]$ is the weight of $(1/\Gamma-1)\Psi(s)$ and $\theta^U=[u_1,u_2,...,u_m]$ is the weight of $(1-1/\Gamma)\Upsilon(s)$.

Moreover, $f^N(s)$ in (19) can be expressed as

$$\begin{aligned} f^N(s) &= \theta^T\Psi(s) + (1/\Gamma-1)\left(\left(\theta^C\right)^T\Psi(s)-\left(\theta^U\right)^T\Upsilon(s)\right) \\ &= f(s) + (1/\Gamma-1)\left(\left(\theta^C\right)^T\Psi(s)-\left(\theta^U\right)^T\Upsilon(s)\right) \end{aligned} \tag{22}$$

The desired value of $(1/\Gamma-1)\left(\left(\theta^C\right)^T\Psi(s)-\left(\theta^U\right)^T\Upsilon(s)\right)$ is$\Delta f^N(s)=f^{N_Tar}-f(s)$, where $f(s)$ is calculated based on (1), and f^{N_Tar} is calculated based on the new trajectory x^N as

$$f^{N\_Tar} = \left(\tau\dot{v}^N - K\left(g-x^N\right) - Dv^N\right)/(g-x_0) \tag{23}$$

From (20) and (21), the old vectors θ and $\Psi(s)$ are preserved in θ^N and $\Psi^N(s)$ and easy to be recovered by reducing new added terms. On the other hand, the skills can keep updating by extending θ^N and $\Psi^N(s)$ by the new features. Equation (22) shows that the new forcing function $f^N(s)$ is calculated based on the old$f(s)$, and we can use f^{N_Tar} and $f(s)$ to compute the desired value of the skill modification $\Delta f^N(s)$ and further to obtain θ^C and θ^U by pseudo-inverse calculations.

Setting the initial value of θ^C as$\Delta f^N(s)(\Psi(s))^+$, the weights θ^C and θ^U are updated by Lemma 1 as

$$\begin{cases} \theta^C = \theta^C - \frac{db^T\Gamma}{1-\Gamma}\Delta f^N(s) \\ \theta^U = \frac{b^T\Gamma}{1-\Gamma}\Delta f^N(s) \\ d = -(\Psi(s))^+\Upsilon(s) \\ c = (\Psi(s))^+\Psi(s)\Upsilon(s) - \Upsilon(s) \\ b^T = \begin{cases} c^+ & \text{if } c\neq 0 \\ \left(1+d^Td\right)^{-1}d^T(\Psi(s))^+ & \text{if } c=0 \end{cases} \end{cases} \tag{24}$$

3.2. Incremental dynamical movement primitive for multiple stylistic skill generalization

IDMP can be applied to generalize multiple stylistic skills when the initial learning skill is not accurate and there are several possible generalization solutions. Considering that all the possible skills are generated based on common initialized features, it is better to allow these skills to share the same features but with different weights. We first assume that the position and velocity terms in the multiple demonstrations are x_i^N and v_i^N, $i=1,2,...,m$ and the variables are x_i and v_i in the initial demonstration. By using (1), we can get an initial skill x_i and v_i as well as the common feature nodes $\Psi(s)$ and the forcing function $f(s)$ in the standard DMP. The next step is to compute the common extended terms $\eta_j(s)$ and the multiple sets of θ^C and θ^U for different trajectories simultaneously.

Here, we set θ_k^C and $\theta_k^U, k=1,2,...,m$ as the vectors of the kth trajectory and set $\Theta^C=\{\theta_1^C,\theta_2^C,...,\theta_m^C\}$ and $\Theta^U=\{\theta_1^U,\theta_2^U,...,\theta_m^U\}$ as vectors of the combination of weights. Similar to (23), we set the ith new learned skill expressed by DMP as

$$\begin{cases} \tau\dot{v}_i^N = K\left(g-x_i^N\right) - Dv_i^N + (g-x_0)f_i^N(s) \\ \tau\dot{x}_i^N = v_i^N \end{cases} \tag{25}$$

The extended term $\eta_j(s)$ and new variable λ_k are computed in the same way as in (11) and (18) to achieve $\Upsilon^c(s)=[\gamma_1(s),\gamma_2(s),...,\gamma_m(s)]$ and$\gamma_i(s)=\eta_i(s)s/\sum_{j=1}^m\eta_j(s)$. Then the ith target value of $f_i^{N\_Tar}(s)$ is

$$f_i^{N\_Tar} = \left(\tau\dot{v}_i^N - K\left(g-x_i^N\right) - Dv_i^N\right)/(g-x_0) \tag{26}$$

and the $f_i^N(s)$ in (25) has a similar expression to $f^N(s)$ in (19) and (22) as

$$\begin{aligned} f_i^N(s) &= \left(\theta_i^N\right)^T\Psi(s) \\ &= \theta^T\Psi(s) + (1/\Gamma^c-1)\left(\theta_i^C\right)^T\Psi(s) + (1-1/\Gamma^c)\left(\theta_i^U\right)^T\Upsilon^c(s) \\ &= f(s) + (1/\Gamma^c-1)\left(\left(\theta_i^C\right)^T\Psi(s)-\left(\theta_i^U\right)^T\Upsilon^c(s)\right) \end{aligned} \tag{27}$$

where Γ^c is defined as same as Γ in (15). The desired value of the term $(1/\Gamma^c-1)\left(\left(\theta_i^c\right)^T\Psi(s)-\left(\theta_i^c\right)^T\Upsilon^c(s)\right)$ in (27) is

$$\Delta f_i^N(s) = f_i^{N\_Tar}(s) - f(s), \tag{28}$$

where $\Upsilon^c(s)$ represents the common extended feature vector, then the term $\theta_i^j, i=1,2,...,m, j=C,U$ in a vector and $\Theta^j, j=C,U$ are calculated based on the common terms $\Upsilon^c(s)$ andΓ^c , similar to $\Upsilon(s)$ and Γ in (19), as

$$\begin{cases} \theta_i^C = \theta_i^C - \frac{db^T\Gamma^c}{1-\Gamma^c}\Delta f_i^N(s) \\ \theta_i^U = \frac{b^T\Gamma^c}{1-\Gamma^c}\Delta f_i^N(s) \\ d = -(\Psi(s))^+\Upsilon^c(s) \\ c = (\Psi(s))^+\Psi(s)\Upsilon^c(s) - \Upsilon^c(s) \\ b^T = \begin{cases} c^+ & \text{if } c\neq 0 \\ \left(1+d^Td\right)^{-1}d^T(\Psi(s))^+ & \text{if } c=0 \end{cases} \end{cases} \tag{29}$$

Using (29), we can get m group vectors of the weight set $\left[\theta_i^C,\theta_i^U\right]$ to express m stylistic skills. The detailed calculation procedure is realized by the pseudo code shown in Algorithm 1, for single and multiple stylistic incremental skill learning.

Algorithm 1 : Incremental Dynamic Movement Primitive

Input: parameters K, D, τ, number of kernels n, Gaussian function ψ_i, $i = 1,2,...,n$, transformation function function $\xi_j(*)$, $j = 1,2,...,m$, start x_0 and end g of trajectory
Demonstrations: 1) x, x^N (single style) or 2) x, x_i^N (multi-style) and convergence condition δ

Output : single style skill learning: return the output weight θ^N (including θ, θ^C and θ^U) and new skill in (9)
multi-style skill learning: return the output weight Θ^N (including θ, Θ^C and Θ^U) and multiple learned skills in (24)

Step 1 : Initial skill learning using DMP
1) Intialize $s, h_i, c_i, i = 1,2,...,n$ and θ and ψ_i;
2) Calculate $f(s)$ by using (4) and f_k^{Tar} in(5)
3) Finalize θ and $\varphi_i(s)$

Step 2 : Calculate terms of the extensive skill features
1) Set the feature mapping group $\Phi(s) = [\varphi_1(s),...,\varphi_n(s)]$
2) **For** $j = 1, j < m$, do
3) Random W_{ej}, β_{ej} and calculate $\eta_j(s) \equiv \zeta_j(\Phi(s)W_{ej} + \beta_{ej})$
4) Calculate Γ in (14) based on $\varphi_i(s)$ and $\eta_j(s)$
5) Calculate $\gamma_j(s)$ in (16) and new feature term λ_k in (17)
6) **End**

Step 3 : New single or multiple stylistic skill learning
1) Initial the basic functions $\Psi(s)$ *and* $\Gamma^c, m = 1$
2) Calculate f^{N_Tar} in (22) and $\Delta f^N(s)$ in (21) for single style skill learning *or* $f_i^{N\_Tar}$ in (25) and $\Delta f_i^N(s)$ in (27) for multi-style skill learning
3) **While** the error term $e = \sum_{i=1}^{k} \left| x - x_i^N \right|$ does't reach the threshold δ, do
4) Random $W_{e(j+1)}, \beta_{e(j+1)}$;
5) Calculate $\eta_{j+1}(s) = \zeta_{j+1}(\Phi(s)W_{e(j+1)} + \beta_{e(j+1)})$ and $\gamma_{j+1}(s)$, add $\gamma_{j+1}(s)$ to extend $\Psi^N(s)$ in (20) and reinitialize θ^N in single style skill learning or Θ^U in multi-style skill learning
5) Use (23) to update θ^C and θ^U and related terms or use (28) to calulate θ_i^C and $\theta_i^U, i = 1,2,...,m$
6) Use $f^N(s) = (\theta^N)^T \Psi^N(s)$ *or* $f_i^N(s) = (\theta_i^N)^T \Psi^N(s)$ to get $f^N(s)$ in single style skill learning or $f_i^N(s)$ in multi-style skill learning
7) $m = m + 1$;
8) **End**
9) Use (9) with $f^N(s)$, or (24) with $f_i^N(s)$ and new starting points, goals and scales to achieve new trajectories x^N and x_i^N

Remark 3. *Since the IDMP-based multi-skill learning are based on the common features, the learned multiple skill can be transformed between each other by only changing the weight vectors. For example, we set the common state vector for two stylistic skills as* $\Psi^N(s) = [\Psi(s)|(1/\Gamma - 1)\Psi(s)|(1 - 1/\Gamma)\Upsilon^c(s)]^T$ *and weight vectors are* $\theta_1^N = \left[\theta\middle|\theta_1^C|\theta_1^U\right]^T$ *and* $\theta_2^N = \left[\theta\middle|\theta_2^C|\theta_2^U\right]^T$ *for two skills with the same length separately. The transformation of the two skills can be realized by linear interpolation* [30] *as.*

$$\begin{cases} \tau \dot{v}^N = K(g - x^N) - Dv^N + (g - x_0^N) f^N(s) \\ \tau \dot{x}^N = v^N \\ f^N(s) = \left(\theta^N\right)^T \Psi^N(s) \\ \theta^N = \alpha\theta_1^N + (1-\alpha)\theta_2^N \end{cases}, \tag{30}$$

where $\alpha \in (0,1)$ *is an adaptive factor to enable* θ^N *to change from* θ_1^N *to* θ_2^N.

4. Incremental adaptive neural network control

The trajectory is replanned using IDMP method in the Section above. However, as the trajectory changes, the controller's setpoints and performance limits also need to be changed. In this Section, we will propose an new incremental adaptive NN control method to accumulate and reuse the learned skill, considering the limitations of tracking errors and robot dynamics estimation errors.

4.1. System dynamics model and control objectives

The dynamic model of robot system is expressed in a Lagrange-Euler form as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + \tau_e \tag{31}$$

where $q \in R^n$ is the simplification of $q(t)$ at time $t \in R^+$ and represents the joint information of robot arm, $M(q) \in R^{n\times n}$ is the inertia matrix, $C(q,\dot{q}) \in R^{n\times n}$ is the Coriolis and centrifugal torque matrix, and $G \in R^n$ is the gravitational torque. The control torques is τ and τ_e is a torques calculated by $\tau_e = J_e^T(q)F_e$, and F_e is forces exerted by the environment, and $J_e^T(q)$ is a Jacobian matrix. Setting the position of the end effector is x, then the relationship of q and x is $x(t) = \phi(q(t)), \dot{x}(t) = J(q(t))\dot{q}(t)$, where $\phi(*)$ is a function for joint and position transformation and $J(q(t))$ is a Jacobian matrix for robot system. The desired value of x is set as x^d, which is achieved by using traditional DMP or I-DMP. Usingx^d, we can calculate q^d and set the tracking error of q to q^d as $e = q^d - q$. The desired tracking performance is to enable e to keep within the predesigned performance$-k_1 v(t) \leqslant e(t) \leqslant k_2 v(t)$, where k_1 and k_2 are constants and $v(t)$ is usually set as an exponential decaying performance function.

The dynamics system satisfies the following properties and assumptions:

Property 1. *The matrices* $\dot{M}(q) - C(q,\dot{q})$ *in* (31) *is skew -symmetric.*

4.2. Incremental adaptive neural network control

In order to realize the predesigned performance, the system controller is designed as

$$\tau = M(q^d)\dot{\alpha} + C\left(q^d, \dot{q}^d\right)\alpha + G(q^d) + K_\tau(\alpha - \dot{q}) - \hat{\tau}_e - g\left(\bar{S}(z)\right) \tag{32}$$

where$\alpha = \dot{q}^d - Le$,and L is a factor calculated in the following equation, K_τ is a positive constant factor, and $\hat{\tau}_e = J_e^T(q)\hat{F}_e$ and $\hat{F}_e$ represents the estimations of τ_e and F_e separately. Usually, the estimation error term $\tilde{\tau}_e = \tau_e - \hat{\tau}_e$ is coupled with uncertainties and disturbances [33,34] to achieve a complex term $Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) = \tilde{M}\dot{\alpha} + \tilde{C}\alpha + \tilde{G} + \tilde{\tau}_e$, where$\tilde{M} = M(q) - M(q^d)$, $\tilde{C} = C(q,\dot{q}) - C\left(q^d, \dot{q}^d\right)$ and $\tilde{G} = G(q) - G(q^d)$ are the uncertain terms caused by joint tracking errors, $\tilde{\tau}_e$ represents the estimation error of the contact torque. $g(\bar{S}(z))$ is an incremental neural networks term with an expression of$g(\bar{S}(z)) = \hat{W}^T\bar{S}(z)$, where $\hat{W}$ is an estimated weight vector and $\bar{S}(z)$ represents a vector consisted of multiple Gaussian functions. We use $g(\bar{S}(z))$ to approach the error term $Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) = W^{*T}\bar{S}(z) + \varepsilon(z)$ that is expressed by the compositions of the desired weight vector W^{*T} and$\bar{S}(z)$, where $\varepsilon(z)$ is the approximation error of the neural network with the limitation of$\| \varepsilon(z) \| \leqslant \varepsilon*, \varepsilon* > 0$.

Taking (32) into (31), we have

$$M(q^d)\dot{\alpha} - M(q)\ddot{q} = -C\left(q^d, \dot{q}^d\right)\alpha + C(q,\dot{q})\dot{q} - K_\tau(\alpha - \dot{q}) - \tau_e + \hat{\tau}_e - G(q^d) - G(q) + g\left(\bar{S}(z)\right) \tag{33}$$

According to definition ofα, we have $\dot{\alpha} = \ddot{q}^d - \dot{L}e - L\dot{e}$ and take it into (33), then

$$\begin{aligned} M(q)(\dot{\alpha} - \ddot{q}) &= -C(q,\dot{q})\alpha + C(q,\dot{q})\dot{q} + K_\tau(\alpha - \dot{q}) + \tilde{M}\dot{\alpha} + \tilde{C}\alpha + \tau_e - \hat{\tau}_e + G(q^d) - G(q) - g\left(\bar{S}(z)\right) \\ &= -C(q,\dot{q})(\alpha - \dot{q}) + K_\tau(\alpha - \dot{q}) + \tilde{M}\dot{\alpha} + \tilde{C}\alpha + \tilde{G} + \tilde{\tau}_e - g\left(\bar{S}(z)\right) \\ &= -C(q,\dot{q})(\alpha - \dot{q}) + K_\tau(\alpha - \dot{q}) + Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) - g\left(\bar{S}(z)\right) \end{aligned} \tag{34}$$

To realize the predesigned performance and ensure the system stability, we set $\delta = \alpha - \dot{q}$ as the velocity-level tracking error to the virtual control term α and build the following barrier Lyaponov function as

$$\begin{aligned} V_q &= V_1 + V_2 \\ &= \sum_{i=1}^{n} h_i \ln\left(\frac{(k_2 v(t))^2}{(k_2 v(t))^2 - e^2}\right) \\ &\quad + \sum_{i=1}^{n} (1-h_i) \ln\left(\frac{(k_1 v(t))^2}{e^2 - (k_1 v(t))^2}\right) + \frac{1}{2}\delta^T M(q)\delta \end{aligned} \tag{35}$$

where $V_2 = \frac{1}{2}\delta^T M(q)\delta$ and V_1 are the items in the rest of (35), and h_i is defined as

$$h_i = \begin{cases} 1 & e > 0 \\ 0 & e \leqslant 0 \end{cases} \tag{36}$$

It is obvious that $V_q > 0$ and the time derivative of V_q is expressed as

$$\begin{aligned} \dot{V}_q &= \frac{1}{2}\sum_{i=1}^{n} h_i \frac{(k_2 v(t))^2 - e^2}{(k_2 v(t))^2} \frac{2k_2 v(t)\dot{v}(t)\left((k_2 v(t))^2 - e^2\right) - (k_2 v(t))^2 (2k_2 v(t)\dot{v}(t) - 2e\dot{e})}{\left((k_2 v(t))^2 - e^2\right)^2} + \\ &\quad \frac{1}{2}\sum_{i=1}^{n} (1-h_i) \frac{e^2 - (k_1 v(t))^2}{(k_1 v(t))^2} \frac{2k_1 v(t)\dot{v}(t)\left(e^2 - (k_1 v(t))^2\right) - (k_1 v(t))^2 (2e\dot{e} - 2k_1 v(t)\dot{v}(t))}{\left(e^2 - (k_1 v(t))^2\right)^2} + \\ &\quad \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T \dot{M}(q)\delta \\ &= \sum_{i=1}^{n} h_i \frac{\dot{v}(t)\left((k_2 v(t))^2 - e^2\right) - k_2 v(t)(k_2 v(t)\dot{v}(t) - e\dot{e})}{k_2 v(t)\left((k_2 v(t))^2 - e^2\right)} + \\ &\quad \sum_{i=1}^{n} (1-h_i) \frac{\dot{v}(t)\left(e^2 - (k_1 v(t))^2\right) - k_1 v(t)(e\dot{e} - k_1 v(t)\dot{v}(t))}{k_1 v(t)\left(e^2 - (k_1 v(t))^2\right)} + \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T \dot{M}(q)\delta \\ &= \sum_{i=1}^{n} h_i \frac{k_2 e\dot{e} v(t) - e^2 \dot{v}(t)}{v(t)\left((k_2 v(t))^2 - e^2\right)} + \sum_{i=1}^{n} (1-h_i) \frac{e^2 \dot{v}(t) - k_1 e\dot{e} v(t)}{v(t)\left(e^2 - (k_1 v(t))^2\right)} + \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T \dot{M}(q)\delta \end{aligned} \tag{37}$$

According to the definition ofα, we have$\dot{q}^d = \alpha + Le = \dot{e} + \dot{q}$, then$\dot{e} = \alpha + Le - \dot{q}$, then

$$\dot{V}_q = \sum_{i=1}^{n} h_i \frac{k_2 e(\alpha + Le - \dot{q})v(t) - e^2\dot{v}(t)}{v(t)\left((k_2 v(t))^2 - e^2\right)} + \sum_{i=1}^{n}(1-h_i)\frac{e^2\dot{v}(t) - k_1 e(\alpha + Le - \dot{q})v(t)}{v(t)\left(e^2 - (k_1 v(t))^2\right)} + \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T\dot{M}(q)\delta$$
$$= \sum_{i=1}^{n} h_i \frac{k_2 e(\alpha - \dot{q})v(t) + k_2 L e^2 v(t) - e^2\dot{v}(t)}{v(t)\left((k_2 v(t))^2 - e^2\right)} + \sum_{i=1}^{n}(1-h_i)\frac{e^2\dot{v}(t) - k_1 e(\alpha - \dot{q})v(t) + Lk_1 e^2 v(t)}{v(t)\left(e^2 - (k_1 v(t))^2\right)} + \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T\dot{M}(q)\delta$$
$$= \sum_{i=1}^{n} h_i \frac{k_2 e(\alpha - \dot{q})}{(k_2 v(t))^2 - e^2} + \sum_{i=1}^{n} h_i \frac{k_2 L e^2 v(t) - e^2\dot{v}(t)}{v(t)\left((k_2 v(t))^2 - e^2\right)} + \sum_{i=1}^{n}(1-h_i)\frac{-k_1 e(\alpha - \dot{q})}{e^2 - (k_1 v(t))^2} + \sum_{i=1}^{n}(1-h_i)\frac{e^2\dot{v}(t) + Lk_1 e^2 v(t)}{v(t)\left(e^2 - (k_1 v(t))^2\right)} + \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T\dot{M}(q)\delta$$
$$= \sum_{i=1}^{n}\left[\frac{h_i k_2 e}{(k_2 v(t))^2 - e^2} - \frac{k_1(1-h_i)e}{e^2 - (k_1 v(t))^2}\right](\alpha - \dot{q}) + \sum_{i=1}^{n} h_i\left(k_2 L - \frac{\dot{v}(t)}{v(t)}\right)\frac{e^2}{(k_2 v(t))^2 - e^2} + \sum_{i=1}^{n}(1-h_i)\left(\frac{\dot{v}(t)}{v(t)} + Lk_1\right)\frac{e^2}{e^2 - (k_1 v(t))^2} + \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T\dot{M}(q)\delta \tag{38}$$

If we set L as$L = -\sqrt{\left(\frac{1}{k_2}\right)^2 + \left(\frac{1}{k_1}\right)^2 + (k_c)^2}\| \frac{\dot{v}(t)}{v(t)} \|$, since $\frac{e^2}{(k_2 v(t))^2 - e^2} > 0$ and$\frac{e^2}{e^2 - (k_1 v(t))^2} > 0$, then we have

$$\sum_{i=1}^{n} h_i\left(k_2 L - \frac{\dot{v}(t)}{v(t)}\right)\frac{e^2}{(k_2 v(t))^2 - e^2} + \sum_{i=1}^{n}(1-h_i)\left(\frac{\dot{v}(t)}{v(t)} + Lk_1\right)\frac{e^2}{e^2 - (k_1 v(t))^2} < -k_c\left(\sum_{i=1}^{n} h_i \frac{e^2}{(k_2 v(t))^2 - e^2} + \sum_{i=1}^{n}(1-h_i)\frac{e^2}{e^2 - (k_1 v(t))^2}\right) \tag{39}$$

Following the inequality $-\frac{e^2}{(k_2 v(t))^2 - e^2} \leqslant -\ln\left(\frac{(k_2 v(t))^2}{(k_2 v(t))^2 - e^2}\right)$ and$-\frac{e^2}{e^2 - (k_1 v(t))^2} \leqslant -\ln\left(\frac{(k_1 v(t))^2}{e^2 - (k_1 v(t))^2}\right)$, (39) can be expressed as

$$\sum_{i=1}^{n} h_i\left(k_2 L - \frac{\dot{v}(t)}{v(t)}\right)\frac{e^2}{(k_2 v(t))^2 - e^2} + \sum_{i=1}^{n}(1-h_i)\left(\frac{\dot{v}(t)}{v(t)} + Lk_1\right)\frac{e^2}{e^2 - (k_1 v(t))^2} < -k_c\left(\sum_{i=1}^{n} h_i \ln\left(\frac{(k_2 v(t))^2}{(k_2 v(t))^2 - e^2}\right) + \sum_{i=1}^{n}(1-h_i)\left(\frac{(k_1 v(t))^2}{e^2 - (k_1 v(t))^2}\right)\right) = -k_c V_1 \tag{40}$$

Then (38) can be simplified as

$$\dot{V}_q \leqslant -k_c V_1 + \sum_{i=1}^{n}\left[\frac{h_i k_2 e}{(k_2 v(t))^2 - e^2} - \frac{k_1(1-h_i)e}{e^2 - (k_1 v(t))^2}\right]\delta + \dot{\delta}^T M(q)\delta + \frac{1}{2}\delta^T\dot{M}(q)\delta \tag{41}$$

According to (34), we have$M(q)\dot{\delta} = -C(q,\dot{q})\delta + K_\tau\delta + Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) - g\left(\bar{S}(z)\right)$. Following Property 1, (41) can be simplified as

$$\dot{V}_q \leqslant \sum_{i=1}^{n}\left[\frac{h_i k_2 e}{(k_2 v(t))^2 - e^2} - \frac{k_1(1-h_i)e}{e^2 - (k_1 v(t))^2}\right]\delta - \delta^T C(q,\dot{q})\delta + \delta^T Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) - \delta^T g\left(\bar{S}(z)\right) + \frac{1}{2}\delta^T\dot{M}(q)\delta - K_\tau\delta^T\delta - k_c V_1$$
$$= \sum_{i=1}^{n}\left[\frac{h_i k_2 e}{(k_2 v(t))^2 - e^2} - \frac{k_1(1-h_i)e}{e^2 - (k_1 v(t))^2}\right]\delta + \delta^T Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) - \delta^T g\left(\bar{S}(z)\right) - K_\tau\delta^T\delta - k_c V_1$$
$$= \left[\sum_{i=1}^{n}\left(\frac{h_i k_2 e}{(k_2 v(t))^2 - e^2} - \frac{k_1(1-h_i)e}{e^2 - (k_1 v(t))^2}\right) + Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) - g\left(\bar{S}(z)\right)\right]\delta - K_\tau\delta^T\delta - k_c V_1 \tag{42}$$

According to the definition of h_i, we have $\frac{h_i k_2 e}{(k_2 v(t))^2 - e^2} \geqslant 0$ and$-\frac{k_1(1-h_i)e}{e^2 - (k_1 v(t))^2} \geqslant 0$, then

$$\max\left(\frac{k_2\| e \|}{\left((k_2 v(t))^2 - e^2\right)}, \frac{k_1\| e \|}{\left(e^2 - (k_1 v(t))^2\right)}\right) \geqslant \frac{h_i k_2 e}{\left((k_2 v(t))^2 - e^2\right)} - \frac{k_1(1-h_i)e}{\left(e^2 - (k_1 v(t))^2\right)} \geqslant 0 \tag{43}$$

then we define $F_i = \frac{h_i k_2 e}{(k_2 v(t))^2 - e^2} - \frac{k_1(1-h_i)e}{e^2 - (k_1 v(t))^2}$ as a positive but limited term.

Then the updating rate of the weight vector $\hat{W}$ is

$$\dot{\hat{W}} = \left(\Gamma_s\bar{S}(z) + u\tanh\frac{u\delta}{\varpi}\right)\delta - K_s\Gamma_s\hat{W} \tag{44}$$

where $\Gamma_s > \| \tilde{W} \|$ is a large positive matrix, $\tanh(*)$ is a hyperbolic tangent function and K_s is positive factor.

We further create a quadratic term $V_m = \tilde{W}^T\Gamma_s^{-1}\tilde{W} > 0$ and the time derivative of V_m is

$$\begin{aligned}\dot{V}_m &= \tilde{W}^T\Gamma_s^{-1}\dot{\tilde{W}} \\ &= \tilde{W}^T\Gamma_s^{-1}\left(\left(-\Gamma_s\bar{S}(z) - u_i\tanh\frac{u_i\delta}{\varpi_i}\right)\delta + K_s\Gamma_s\hat{W}\right) \\ &= \tilde{W}^T\Gamma_s^{-1}\left(\left(-\Gamma_s\bar{S}(z) - u_i\tanh\frac{u_i\delta}{\varpi_i}\right)\delta + K_s\Gamma_s\left(W^* - \tilde{W}\right)\right) \\ &= -\tilde{W}^T\bar{S}(z)\delta - \tilde{W}^T\Gamma_s^{-1}u_i\tanh\frac{u_i\delta}{\varpi_i}\delta + \tilde{W}^T K_s\left(W^* - \tilde{W}\right)\end{aligned} \tag{45}$$

Therefore for the hybrid Lyapunov function $V = V_q + V_m > 0$, the time derivative is expressed as

$$\dot{V} \leqslant \left[\sum_{i=1}^{n} F_i + Y\left(q,\dot{q},e,\dot{e},\tilde{\tau}_e\right) - g\left(\bar{S}(z)\right)\right]\delta - \tilde{W}^T\bar{S}(z)\delta - \tilde{W}^T\Gamma_s^{-1}u_i\tanh\frac{u_i\delta}{\varpi_i}\delta + \tilde{W}^T K_s\left(W^* - \tilde{W}\right) - K_\tau\delta^T\delta - k_c V_1$$
$$= \left[\sum_{i=1}^{n} F_i - \tilde{W}^T\Gamma_s^{-1}u_i\tanh\frac{u_i\delta}{\varpi_i} + \varepsilon(z)\right]\delta + \tilde{W}^T K_s\left(W^* - \tilde{W}\right) - K_\tau\delta^T\delta - k_c V_1 \tag{46}$$

Following the Young's inequality and definition ofΓ_s, we can obtain the following inequality [18]

$$F\delta - \tilde{W}^T\Gamma_s^{-1}u\delta\tanh\frac{u\delta}{\varpi} \leqslant \imath\varpi \tag{47}$$

Thus (46) can be further deduced by expressing the weights by the extended matrices as $W^* = \left[W^*_{ori} \;\; W^*_{ex}\right]^T$ and$\tilde{W} = \left[\tilde{W}_{ori} \;\; \tilde{W}_{ex}\right]^T$, where W^*_{ori} and $\tilde{W}_{ori}$ represent the vectors of the original weight and weight error, and W^*_{ex} and $\tilde{W}_{ex}$ are the vectors of the extend features. Then

$$\dot{V} \leqslant -k_c V_1 + \frac{1}{2}\varepsilon(z)^2 + \imath\varpi - K_\tau\delta^T\delta - \frac{1}{2}\tilde{W}^T K_s\tilde{W} + \frac{1}{2}W^{*T}K_s W^*$$
$$= -k_c V_1 - K_\tau\delta^T\delta - \frac{1}{2}K_s\left\| \begin{matrix}\tilde{W}_{ori} \\ \tilde{W}_{ex}\end{matrix} \right\|^2 + \frac{1}{2}\varepsilon(z)^2 + \imath\varpi + \frac{1}{2}K_s\left\| \begin{matrix}W^*_{ori} \\ W^*_{ex}\end{matrix} \right\|^2 \tag{48}$$

Considering the completed expression of V is $V = V_1 + V_2 + V_m = V_1 + \delta^T M(q)\delta + \tilde{W}^T\Gamma_s^{-1}\tilde{W}$, then can be expressed as

$$\dot{V} = -\eta V + \sigma \tag{49}$$

where $\eta = \min\left(\lambda_{\min}(k_c), \frac{2\lambda_{\min}(K_\tau)}{\lambda_{\max}(M(q))}, \frac{2\lambda_{\min}(K_s)}{\lambda_{\max}\left(\Gamma_s^{-1}\right)}\right)$ and$\sigma = \frac{1}{2}\varepsilon(z)^2 + \imath\varpi + \frac{1}{2}K_s\left\| \begin{matrix}W^*_{ori} \\ W^*_{ex}\end{matrix} \right\|^2$, and the solution is

$$V(t) \leqslant \left(V(0) - \frac{\sigma}{\eta}\right)\exp\left(-\eta t\right) + \frac{\sigma}{\eta} \leqslant V(0)\exp\left(-\eta t\right) + \frac{\sigma}{\eta} \qquad (50)$$

Since the terms $\varepsilon(z)$, $\iota\varpi$ and $K_s\| \left[\tilde{W}^{*}_{ori}\ \tilde{W}\right]^T \|^2$ are bounded and σ/η is bounded, then $V(t)$ is bounded and converged along with the time. This completes the proof.

Remark 4. *In our previous research [18] and [33], we proposed a combining framework of trajectory learning and board learning-based control to approximate the unknown dynamics of the robot. The improvements of this proposed method are building a new Lyapunov function and creating a new weight estimation function (44) based on the trajectory learned by IDMP. Therefore, the controller is designed with a specific parameters as L in the definition of* α.

5. Experiments

We will verify the three contributions shown in the Introduction through three following experiments.

5.1. Experiment 1: Accurate trajectory approaching

In this experiment, we aim to verify the improvement of trajectory tracking accuracy by IDMP, compared with other DMP-based methods. We prepare a handwriting letter ‘A’ in blue in Fig. 3 and use the standard DMP, DMP + proposed by Wang and Wu et al. in [29], and IDMP in this paper to track this trajectory with the same kernels.

First we choose 5 kernels for the forcing function for all DMP-based methods. The 5 initial kernels are chosen with random centres and widths of the radial basis functions for all three methods. The initial learning results of DMP in Fig. 3(a) show a not good tracking effect.

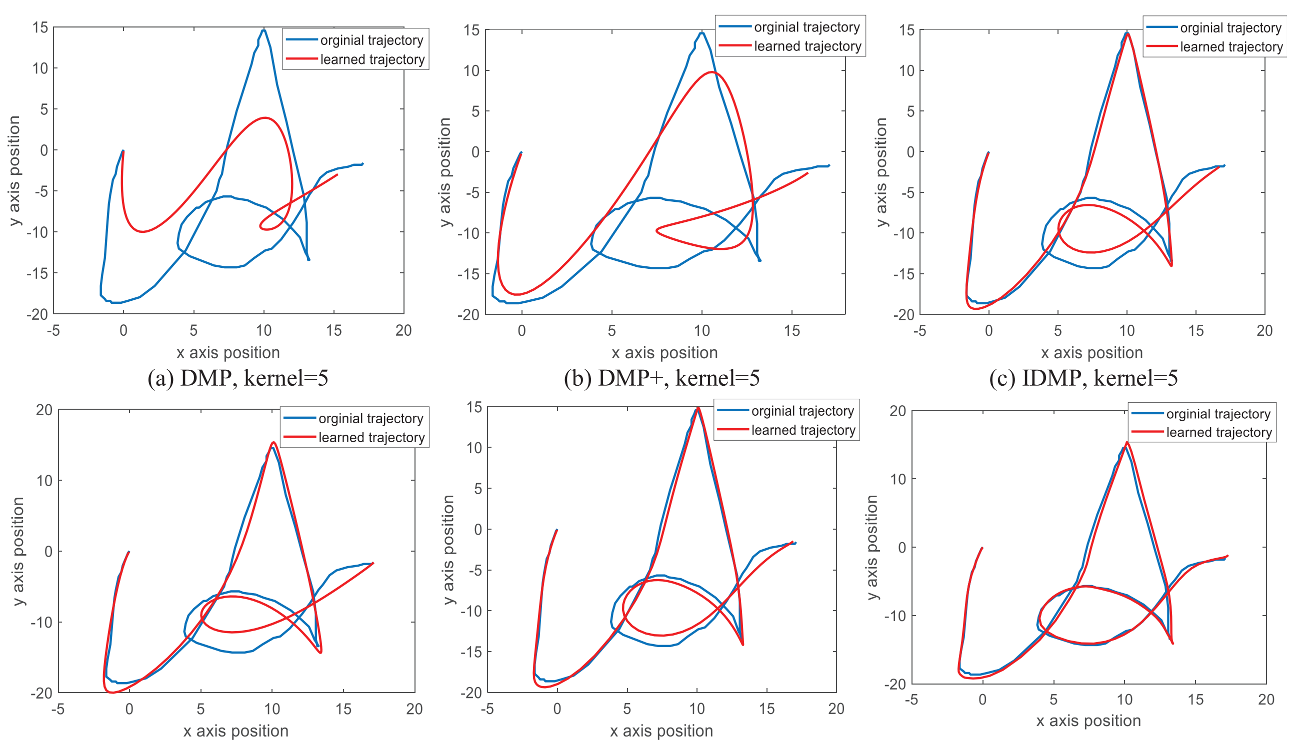
With DMP+, the 5 kernels are modified and the mean squared error (MSE) in tracking the demonstrated trajectory is significantly reduced (see Fig. 3(b)). IDMP can extend the feature vector $\Psi(s)$ by adding new transformation terms $\eta_j(s)$ of the 5 original kernels in (11), so that the trajectory tracking performance is further improved (see Fig. 3(c)).

We further expand the number of initial kernels in three methods from 5 to 20 and the simulation results are shown in Fig. 3 (d) to (f). The tracking performances of all methods are significantly better than those with 5 kernels. The MSEs to the original trajectory of IDMP are much lower than the results of the previous two cases and the trajectory almost coincides with the demonstration, which benefits from the increasing number of the extended features and certifies that IDMP has the best trajectory tracking accuracy among the three methods. But, it also shows that the tracking errors are still partially affected by the initial number of kernels.

5.2. Experiment 2:Multi-style skill learning and transformation

The second experiment is to examine multi-style skill learning, modification, and transformation. As shown in Fig. 4, we retrofitted a PHANTOM Desktop haptic device and fixed a pen at the end of the effector. The demonstrator operates the haptic device to write letters and the device records trajectories of the end tip. The trajectories are processed (e.g., alignment and filtering) and then used for handwriting style learning and transformation under the control of the incremental adaptive NNcontroller in (32).

As shown in Fig. 5 (a), we write five letters ‘a’, ‘z’, ‘l’, ‘k’ and ‘w’ with similar size and same start and end. The letter’a’ is selected as the initial writing style and the others are provided as the writing



(a) DMP, kernel=5 (b) DMP+, kernel=5 (c) IDMP, kernel=5
(d) DMP, kernel=20 (e) DMP+, kernel=20 (f) IDMP, kernel=20

Fig. 3. Handwritting trajectory learning by using standard DMP, DMP + and IDMP with different kernals.

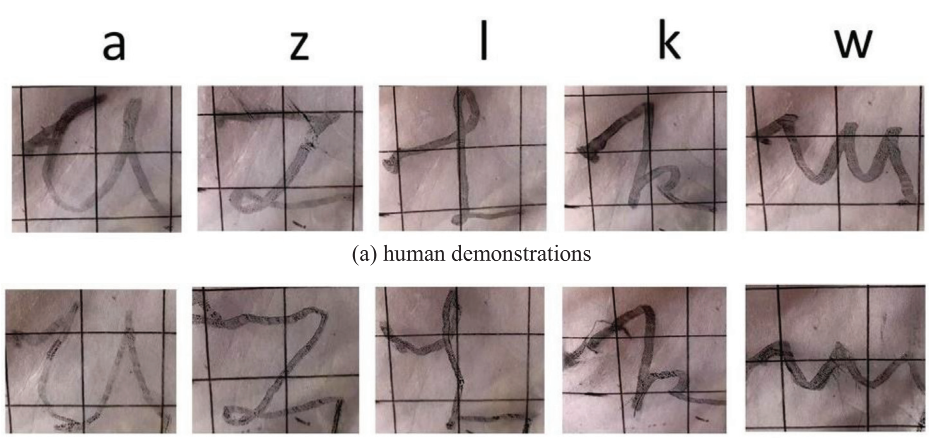
Fig. 4. Experimental setup.

styles after skill modification and transformation. After matching and resampling the demonstrated trajectories, we use the method described in **Algorithm 1** to learn stylistic letters and realize skill modification and transformation from ‘a’ to other letters. The processes for the skill transformations are presented in Fig. 6.

The initial trajectory ‘a’ is coloured red, marked with a red square in the centre of Fig. 6, and learned with the standard DMP. The targeted manuscripts are presented with black squares in the four corners. The transformation starts at centre ‘a’ to approach the handwritings in the corners, performing every 5 incremental steps. In this way, the shapes near the centre are more similar to ‘a’. With the extension of feature vector and weight vector, the learned trajectories are gradually changed from ‘a’ to other stylistic letters. After adding 30 extended feature kernels, the modified letters are clearly distinguishable from each other, resulting in the letters in the corners of Fig. 6. For some letters, such as ‘z’ and ‘k’, the modified letters from ‘a’ have been similar to the final trajectories. While, others, such as ‘w’ and ‘l’, are changed gradually to the desired states. Since the common features are determined by all the stylistic letters, the trained trajectories that are closest to the corner demonstrations still have some differences that can be considered as compromise results. The skill transformation between different stylistic actions is realized by using (30) to change weight vectors. Finally, we utilize the haptic device as an actuator and use adaptive NN control method to draw the learning results, as shown in Fig. 5(b).

5.3. Experiment 3: Dual-incremental skill learning and control for crossing different-height obstacles

The third experiment is also conducted with the haptic device PHANTOM Desktop and a height-adjustable obstacle to illustrate incremental skill learning process and its application of obstacle avoidance in practice. As shown in Fig. 7, humans hold the joystick to cross the obstacle and put the pen tip to touch the intended target point. The current obstacle is consists of three 2 cm*2 cm*2 cm cubes, each of which can be added and removed to change the height of the obstacle. On the top of the cubes, we add a 2 cm*8 cm*0.2 m lip. On the base plane, we set one start point and nine target points on the two sides of the obstacle. The central point on the right side is used for human demonstration and the other points, which are 2 cm away from each other, are used for skill generalization.



(a) human demonstrations

(b) robot drawing results

Fig. 5. Human demonstrations of handwritting and robot drawing results after multi-style skill learning.

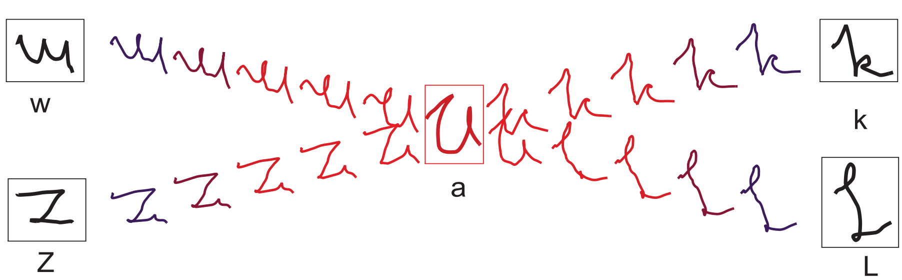


Fig. 6. Incremental learning process from the letter ‘a’ to multilple syles of handwrittings:’w’,’z’,’k’and’l’.

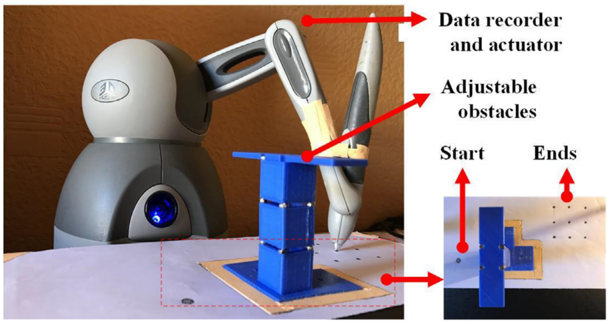


Fig. 7. Experimental setup for obstacle avoidance.

After collecting data from demonstrations (gray lines in Fig. 8 (a)), we use DMP to learn an initial skill of crossing an obstacle with a height of 1 block (red line in Fig. 8 (a)). In our previous work [21,28], the demonstrations for the height-adjustable obstacle are divided into several phases and the final positions of the internal phases are changed according to the heights of the new obstacles. In this paper, we use the IDMP to realize skill modification. As shown in Fig. 8 (b), the deep green and dark blue lines are the final learning results for crossing two and three blocks. The lines with gradient colours from red to green and from green to blue show the learning process with the increase of extended features, according to the diagram in Algorithm 1. Fig. 8 (b) also verifies the convergence of the learning results such that the degrees of curve changes are decreasing until they approach the final learning results. After gaining the ability to overcome the obstacles of different heights, we can generalize them to achieve different goals by changing factor g in (25).

In Fig. 8 (c) and (d), the green lines show the learning effect for different obstacles and the red lines are the generalized trajectories of the robot end to achieve different goals.

Using the adaptive NN controller in (32), the joystick works as an actuator to follow the generalized skills with limited tracking errors. Figure. 9 (a) shows human demonstrations process. Figure. 9 (b) and Figure. 9 (c) show the joystick movement to reach the predefined goals without conflicting with obstacles.

5.4. Discussion

The three experiments verify three properties of the incremental trajectory and force learning framework for robots proposed in the Introduction. Experiment 1 shows the advantage of IDMP in terms of accuracy, compared to DMP + and standard DMP and

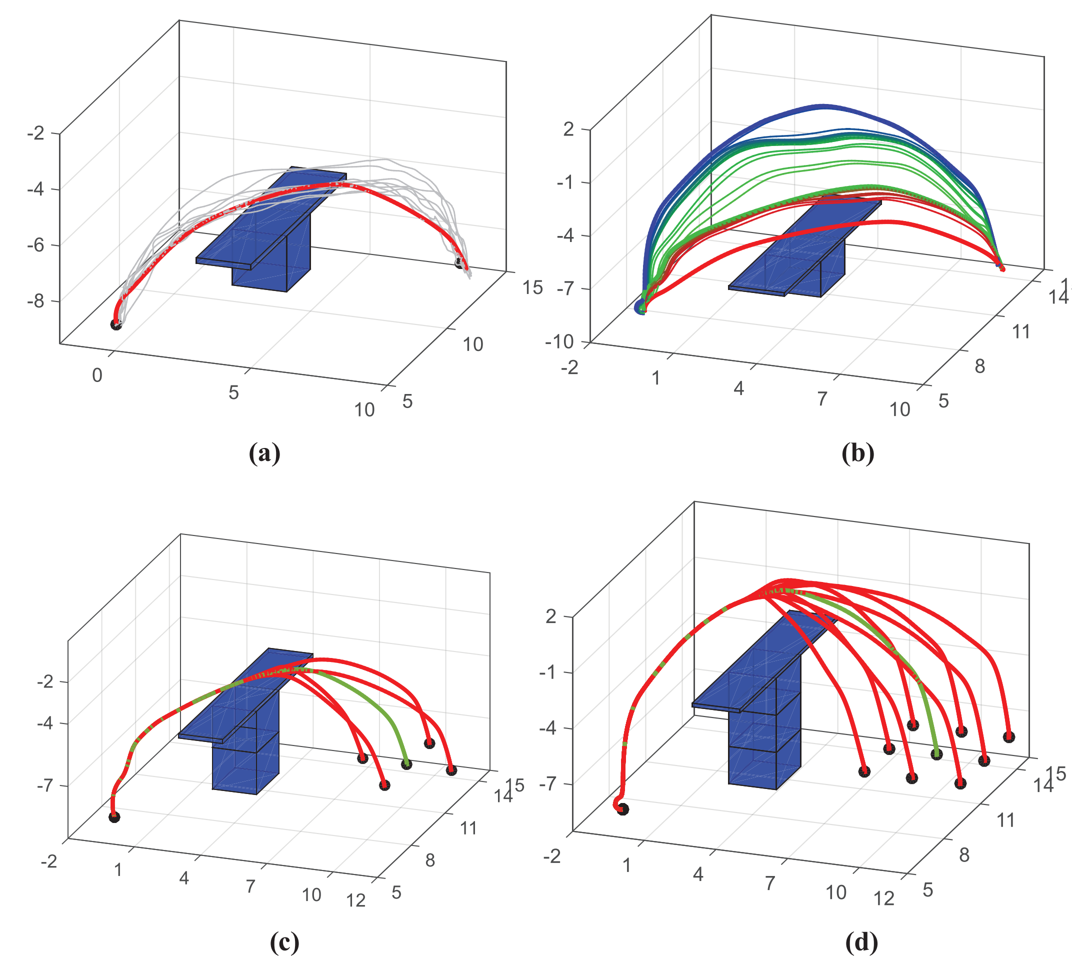
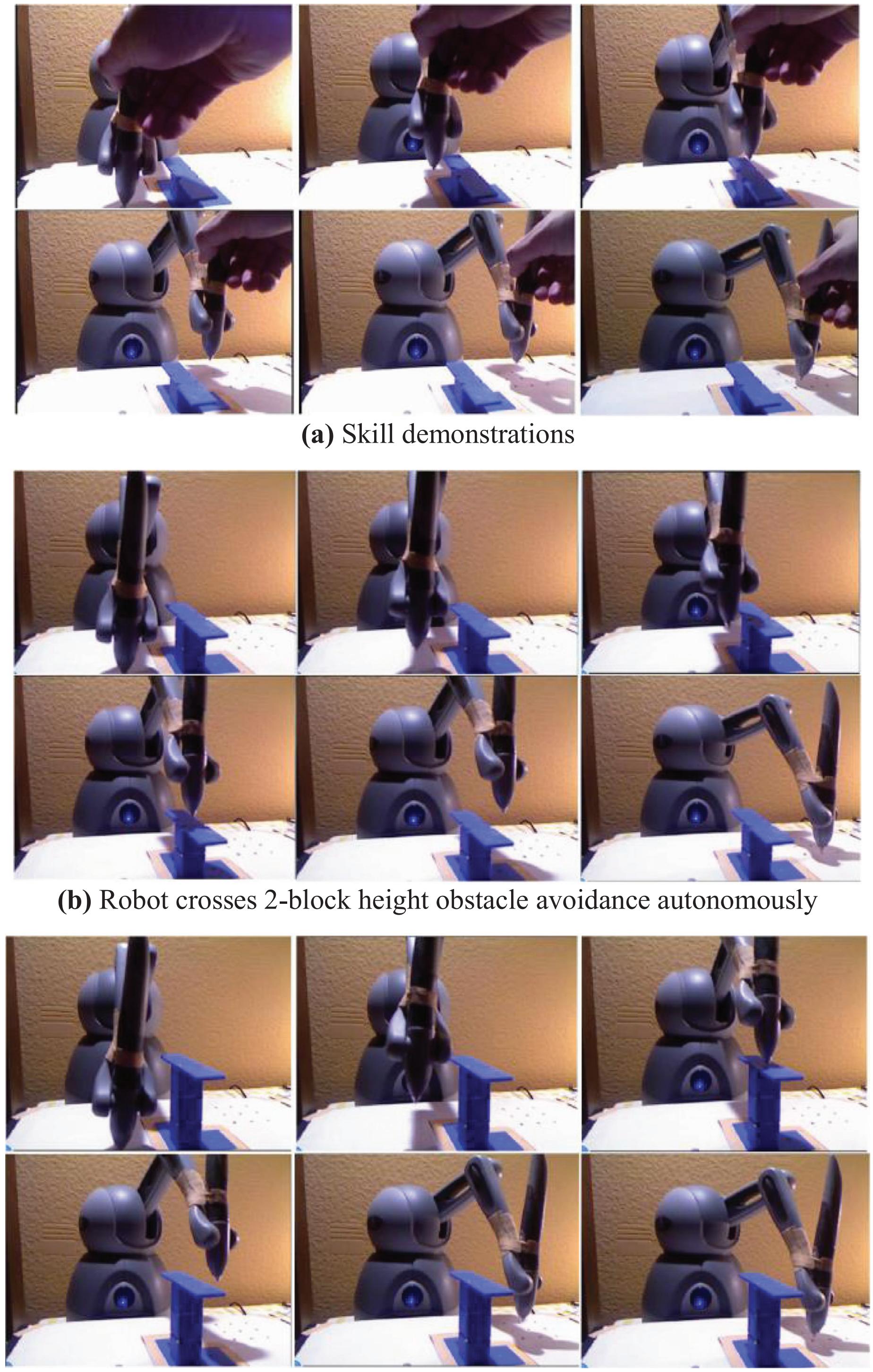


Fig. 8. Skill incremental learning and generalization based on IDMP and adaptive NN-based control method (a) Demonstrations and skill learning based on DMP (gray lines are trajectories of demonstrations and the red line is the learned skill) (b) Incremental learning process for different learning skills (red line is the crossing skill for 1 block height, green line is the crossing skill for 2-block height, and blue line is for the height of 3 blocks, and the thin lines between the different skills represent the transformation of the skills) (c) Generalization of the skill for the 2-block height (green line shows the skill learning and red lines represent the skill generalizations) (d) Generalization of the skill for the height of 3 blocks (green line shows the skill learning and red lines show the skill generalization).



(a) Skill demonstrations

(b) Robot crosses 2-block height obstacle avoidance autonomously

(c) Robot crosses 3-block height obstacle avoidance autonomously

Fig. 9. Skill generalization and control for height-adjustable obstacle avoidance based on human demonstations.

can realize life-long skill learning to some extent. Experiment 2 shows the versatility of IDMP in generalizing and transforming skills into multiple styles. The incremental adaptive controller ensures system stability and keeps trajectory tracking errors within performance limits. Experiment 3 shows applications of the proposed framework in obstacle avoidance. Moreover, the properties can be used in combination to achieve other goals. For example, we can first generalize the original skill to multi-style skills, and then further refine the details of a specific style. Since the old features and weights are contained in the vectors, we can easily perform skill transformation from one to another or transformation between different skills as Remark 3 shown, and ensure the smoothness of the transformation by choosing an appropriateα. But, since IDMP is calculated based on the elements of the original DMP skills, the proposed method is still limited by the original learning outcomes.

6. Conclusion

In this paper, we propose a new framework for incremental trajectory and force learning and generalization to modify initially learned skills due to the environmental changes and inaccurate initialization. The trajectory learning part is based on the IDMP, and the controller uses adaptive NN control method to reduce the cost and improve the efficiency of learning. Three experiments are taken to verify the effectiveness and advantages of the proposed framework in accurate trajectory tracking, multi-stylistic trajectory tracking and application in obstacle avoidance. Compared to other DMP-based methods, this framework achieves better performance in robot trajectory tracking and greater flexibility in skill modification and transformation.

CRediT authorship contribution statement

Zhenyu Lu: Conceptualization, Methodology. **Ning Wang:** Software. **Qinchuan Li:** Project administration and Reviewing. **Chenguang Yang:** Supervision, Project administration.

Data availability

The authors are unable or have chosen not to specify which data has been used.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/S001913, in part by the H2020 Marie Skłodowska-Curie Actions Individual Fellowship under Grant 101030691, and in part by the Key Research and Development Project of Zhejiang Province under Grant number 2021C04017.

References

[1] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Rob. Auton. Syst. 57 (5) (2009) 469–483.
[2] H. Ravichandar, A.S. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration, Annu. Rev. Control, Robot., Autonomous Syst. 3 (2020) 297–330.
[3] Y. Huang, L. Rozo, J. Silvério, D.G. Caldwell, Kernelized movement primitives, Int. J. Robot. Res. 38 (7) (2019) 833–852.
[4] M. Deniša, T. Petric, A. Gams, A. Ude, A review of compliant movement primitives, Robot Control (2016) 1–17.
[5] M. Saveriano, F.J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey. arXiv preprint arXiv:2102.03861 (2021).
[6] S. Schaal, J. Peters, J. Nakanishi, A. Ijspeert, Control, planning, learning, and imitation with dynamic movement primitives, in: Workshop on Bilateral Paradigms on Humans and Humanoids: IEEE International Conference on Intelligent Robots and Systems (IROS 2003), 2003, pp. 1–21.
[7] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, Neural Comput. 25 (2) (2013) 328–373.
[8] Y. Yuan, Z. Li, T. Zhao, D. Gan, DMP-based motion generation for a walking exoskeleton robot using reinforcement learning, IEEE Trans. Ind. Electron. 67 (5) (2019) 3830–3839.
[9] Z. Li, T. Zhao, F. Chen, Y. Hu, C.Y. Su, T. Fukuda, Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoid like mobile manipulator, IEEE/ASME Trans. Mechatron. 23 (1) (2017) 121–131.
[10] A. Pervez, Y. Mao, D. Lee, Learning deep movement primitives using convolutional neural networks, in: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), 2017, pp. 191–197.
[11] C. Yang, C. Chen, W. He, R. Cui, Z. Li, Robot learning system based on adaptive neural control and dynamic movement primitives, IEEE Trans. Neural Networks Learn. Syst. 30 (3) (2018) 777–787.
[12] T. Kastritsi, F. Dimeas, Z. Doulgeri, Progressive automation with dmp synchronization and variable stiffness control, IEEE Rob. Autom. Lett. 3 (4) (2018) 3789–3796.
[13] C. Yang, C. Chen, N. Wang, Z. Ju, J. Fu, M. Wang, Biologically inspired motion modeling and neural control for robot learning from demonstrations, IEEE Trans. Cognit. Dev. Syst. 11 (2) (2018) 281–291.
[14] F.J. Abu-Dakka, V. Kyrki, Geometry-aware dynamic movement primitives, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 4421–4426.
[15] H. Hoffmann, P. Pastor, D.H. Park, S. Schaal, Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance, in: 2009 IEEE International Conference on Robotics and Automation, 2009, pp. 2587–2592.
[16] J. Umlauft, D. Sieber, S. Hirche, Dynamic movement primitives for cooperative manipulation and synchronized motions, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 766–771.
[17] A. Gams, B. Nemec, A.J. Ijspeert, A. Ude, Coupling movement primitives: Interaction with the environment and bimanual tasks, IEEE Trans. Rob. 30 (4) (2014) 816–830.
[18] H. Huang, T. Zhang, C. Yang, C.P. Chen, Motor learning and generalization using broad learning adaptive neural control, IEEE Trans. Ind. Electron. 67 (10) (2019) 8608–8617.
[19] Z. Liu, J. Zhou, C.P. Chen, Broad learning system: Feature extraction based on K-means clustering algorithm, in: 2017 4th International Conference on Information, Cybernetics and Computational Social Systems (ICCSS), 2017, pp. 683–687.
[20] C.P. Chen, Z. Liu, Broad learning system: An effective and efficient incremental learning system without the need for deep architecture, IEEE Trans. Neural Networks Learn. Syst. 29 (1) (2017) 10–24.
[21] C. Yang, C. Zeng, Y. Cong, N. Wang, M. Wang, A learning framework of adaptive manipulative skills from human to robot, IEEE Trans. Ind. Inf. 15 (2) (2018) 1153–1161.
[22] L. Han, P. Kang, Y. Chen, W. Xu, B. Li, Trajectory optimization and force control with modified dynamic movement primitives under curved surface constraints, in: 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2019, pp. 1065–1070.
[23] Z. Lu, N. Wang, C. Yang, A novel iterative identification based on the optimised topology for common state monitoring in wireless sensor networks, Int. J. Syst. Sci. 53 (1) (2022) 25–39.
[24] R. Huang, H. Cheng, J. Qiu, J. Zhang, Learning physical human–robot interaction with coupled cooperative primitives for a lower exoskeleton, IEEE Trans. Automation Sci. Eng. 16 (4) (2019) 1566–1574.
[25] M. Saveriano, D. Lee, October). Incremental skill learning of stable dynamical systems, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 6574–6581.
[26] A. Lemme, R.F. Reinhart, J.J. Steil, Self-supervised bootstrapping of a movement primitive library from complex trajectories, in: 2014 IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 726–732.
[27] Y. Wu, R. Wang, L.F. D'Haro, R.E. Banchs, K.P. Tee, October). Multi-modal robot apprenticeship: Imitation learning using linearly decayed DMP+ in a human-robot dialogue system, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1–7.
[28] C. Yang, C. Zeng, C. Fang, W. He, Z. Li, A DMPs-based framework for robot learning and generalization of humanlike variable impedance skills, IEEE/ASME Trans. Mechatron. 23 (3) (2018) 1193–1203.
[29] R. Wang, Y. Wu, W.L. Chan, K.P. 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.
[30] S. Calinon, Gaussians on Riemannian Manifolds: Applications for Robot Learning and Adaptive Control, IEEE Robot. Automation Mag. 27 (2) (2020) 33–45, https://doi.org/10.1109/MRA.2020.2980548.
[31] S. Schaal, Dynamic Movement Primitives-a Framework for Motor Control in Humans and Humanoid Robotics, in: Adaptive motion of animals and machines, Springer, Tokyo, 2006, pp. 261–280.
[32] N. Wang, C. Chen, C. Yang, A robot learning framework based on adaptive admittance control and generalizable motion modeling with neural network controller, Neurocomputing 390 (2020) 260–267.
[33] Z. Liu, W. Lin, X. Yu, J.J. Rodriguez-Andina, H. Gao, Approximation-Free Robust Synchronization Control for Dual Linear Motors Driven Systems With Uncertainties and Disturbances, IEEE Trans. Ind. Electron. (2021), In express.
[34] P. Shi, W. Sun, X. Yang, I. Rudas, H. Gao. Master-Slave Synchronous Control of Dual Drive Gantry Stage with Cogging Force Compensation. IEEE Transactions on Systems, Man, and Cybernetics: Systems. doi: 10.1109/TSMC.2022.3176952.
[35] Y. Zhang, M. Li, C. Yang, Robot learning system based on dynamic movement primitives and neural network, Neurocomputing 451 (2021) 205–214.
[36] W. Si, N. Wang, C. Yang, Composite dynamic movement primitives based on neural networks for human–robot skill transfer, Neural Comput. Appl. (2021) 1–11.
[37] T. Matsubara, S.H. Hyon, J. Morimoto, Learning stylistic dynamic movement primitives from multiple demonstrations, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 1277–1283.

[38] M. Akbulut, E. Oztop, M.Y. Seker, X. Hh, A. Tekden, E. Ugur, Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing, in: Conference on Robot Learning, PMLR, 2021, pp. 1896–1907.

Zhenyu Lu received the Ph.D degree in Northwestern Polytechnical University, Xi'an, China in 2019. He is currently working as a senior research fellow in Bristol Robotic Laboratory & University of the West of England, Bristol. His research interests include teleoperation, human–robot interaction and intelligent learning method.

Ning Wang is a Senior Lecturer in Robotics at the Bristol Robotics Laboratory, University of the West of England, United Kingdom. She received the M.Phil. and Ph.D. degrees in electronics engineering from the Department of Electronics Engineering, The Chinese University of Hong Kong, Hong Kong, in 2007 and 2011, respectively. Her research interests lie in signal processing, intelligent data analysis, human-robot interaction and autonomous driving.

Prof. Qinchuan Li received the B.S. degree in fluid power transmission and control from Yanshan University, Hebei province, China, in 1997, and the Ph.D. degrees in mechanism design and theory from Yanshan University in 2003. He is currently a professor and director of Robotics Lab at Zhejiang Sci-Tech University, Hangzhou, China. His research interests include robotics and mechanism design, particularly parallel manipulators, covering topology design, kinematics, dynamics and some applications.

Prof. Chenguang Yang received the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010, and postdoctoral training in human robotics from the Imperial College London, London, U.K. He was awarded UK EPSRC UKRI Innovation Fellowship and individual EU Marie Curie International Incoming Fellowship. As the lead author, he won the IEEE Transactions on Robotics Best Paper Award (2012) and IEEE Transactions on Neural Networks and Learning Systems Outstanding Paper Award (2022). He is the Corresponding Co-Chair of IEEE Technical Committee on Collaborative Automation for Flexible Manufacturing, a Fellow of Institute of Engineering and Technology (IET), and a Fellow of British Computer Society (BCS). His research interest lies in human robot interaction and intelligent system design.