A Novel Human-robot Skill Transfer Method for Contact-rich Manipulation Task

Jiale Dong, Weiyong Si, Chenguang Yang*

Structured abstract

Learning from Demonstration (LfD) has been proven an effective method for robots to learn skills from humans, avoiding time-consuming programming by robot experts and allowing users easily transfer skills to robots through intuitive demonstration. However, traditional LfD often lacks good generalisation ability for new conditions, which may lead to task failure when the external environment changes or human perturbations arise, especially for complex tasks with multiple steps in practice. To tackle this, we improve the original dynamic motion primitives (DMPs), which significantly increases the reusability and generalisation of skills. In addition, we propose a behavior tree(BT)-based framework for robots to learn multi-step complex tasks. After acquiring primitive motion skills through demonstration, the robot is able to select the appropriate motion primitives based on the perception state to complete the task when tasks and environments change. To further improve the generalisation, we enhance the framework by combining it with the broad learning system(BLS), in which the BT generates the robot's motion trajectories, and BLS optimises motion parameters further. To verify the effectiveness of the proposed method, we carried out experiments on the Baxter robot and the elite robot through wiping tables and assembling components, respectively. The results of these experiments confirmed the effectiveness of the proposed method.

Keywords: Skill learning, Broad learning system, Behavior tree, Assembling, Dynamic motion primitives.

I. Introduction

As robots are expected to be applied in the real world, learning from demonstration (LfD) has become a common and popular method for robots to acquire new skills (Si et al., 2021*b*, Pignat et al., 2022). LfD can learn trajectory skills from human demonstrations, and its complexity is lower compared to planning methods (Liao et al.,2021, Zhang et al., 2014, Liao & Liu, 2015). Nowadays, LfD has been widely used in various industries, such as manufacturing and medical examination. One advantage of LfD is that it does not require the human teacher to own expertise in robotics and programming, the user only needs to demonstrate how to perform a task such as by dragging robots to record the robots' states. And the robot can acquire human skills through learning from human demonstrations (Ravichandar et al., 2020). Thus, it is convenient to learn manipulation skills from a few demonstrations, making it a user-friendly method for robotic programming in practice. However, the LfD method still needs to be improved due to the limited generalisation capability. The existing LfD method usually only learns the motor skills taught by human beings, but can not well learn human perception and decision-making ability, which is also one of the core issues in the field of robotics (Qiao et al., 2022, Qiao et al., 2023).

The application of robots in complex sequential actions has gained increasing attention from robotic researchers (Hauge et al., 2021). For unknown environments and complex tasks, humans can easily reuse previously learned skills by selecting and adjusting them to improve their performance. However, it is difficult for robots to re-teaching and re-program when the environment and tasks change. To this end, we can decompose complex skills into multiple sub-skills and then complete complex operations through their combinations (Si et al., 2022). Sub-skills can be represented by motion primitives, and common modelling methods include dynamic motion primitives (DMPs) (Ijspeert et al., 2001) and probabilistic models (Calinon & Billard, 2009, Rozo et al., 2011). The essence of DMPs is a second-order nonlinear equation, which allows changing the final state, speed and duration of the motion without changing the overall shape of the motion, and can easily expand and adjust the motion trajectory (Pastor et al., 2009, Ijspeert et al., 2013). Nowadays, DMPs have been widely used in various fields of robotics (Yang et al., 2018, Han et al., 2022), and some researchers have improved the original DMPs. In their work (Prada et al., 2013), the authors presented a method to decouple nonlinear terms from phase, enabling robots to adapt to human behavior when performing tasks that involve human interaction. Subsequently, (Wang et al., 2016, Wu et al., 2018) proposed an efficient trajectory connection and insertion technique by eliminating the problem of exponential time decay disappearance, whereas (Kulvicius et al., 2011) introduced a method for smooth transition between multiple DMPs in position and velocity space. Building upon these contributions, this paper presents a modified DMP model that allows for rotation in space and scaling of shape to improve the generalization performance of DMPs in different tasks. The aim of this research is to enhance the reusability and applicability of DMPs.

In addition, finite state machine (FSM) is often used to combine motion primitives to construct complex behaviors (Bohren & Cousins, 2010), but this usually leads to systems being too complex and large, and difficult to extend and reuse. Behavior trees (BTs) can solve such problems very well. BTs have the advantages of easy expansion and high reusability. In recent years, it has received great attention in the field of robotics (Iovino et al., 2020, Styrud et al., 2022). BT enables robots to perform corresponding actions under specific conditions, but how to do it is still a problem. Similar to the human learning process, our humans can easily learn the action arrangement of a task, but how much force each action needs to use, and where it needs to be moved, which requires a learning process. Take shooting as an example, there are only two movements in shooting: lifting the ball and throwing the ball. However, if you want to successfully throw the ball in, you need to lift the ball to the appropriate pose and shoot with the appropriate force, which requires a lot of practice. By imitating the human learning process, we combine BT with broad learning system (BLS), make motion decisions with BT, generate specific motion parameters with BLS.

In recent years, some researchers have applied various advanced machine learning methods to robot skills learning. However, these methods usually require huge data to train models, and it is not feasible or expensive to collect these data on real robots. In this process, the robot needs to interact with the environment, and it is difficult to ensure the safety of the interaction between the robot and the environment. A feasible method is to execute the training process in a simulated environment, and then transfer the training model to a real robot, which is called transfer learning. However, it is still difficult to build a high-fidelity simulator to train skills. (Lončarević et al., 2021) proposed a statistical generalization method to generate database based on real data, and verified the effectiveness of data in

the pitching task.

To solve the above problems, we propose a framework that combines BT, LfD and BLS. First, the robot obtains a group of motion primitives through LfD. Human constructs a BT for specific tasks to build a motion planner for robots. And BLS generates specific parameters of the motion. Compared with other machine learning methods applied to robots, BLS in this method is only responsible for the generation of motion parameters, thus its training cost is very low. This framework has the advantages of generalization, anti-interference, and only a small amount of training data. We will explain the proposed methods in Section III.

The main contributions of this paper are as follows:

1) A human-robot skill transfer framework by combining LfD, BTs and BLS is proposed, and the effectiveness of the proposed method is proved in the tabletop cleaning and peg-ring assembly task.

2) We modify the original DMPs model so that it can scale the shape of the trajectory, and combine it with the DMPs model that can be rotated, which improves the reusability and applicability of DMPs.

3) The motion skills and controller character are encapsulated into the action nodes of BT, which enhances the learning and generalization performance of the robot.

The rest of the article is structured as follows. In Section II, the proposed method, including BTs, DMPs and BLS are presented. Section III presents two experiments to evaluate the effectiveness of the proposed method. Section IV summarizes the contribution of this paper and analyzes the advantage of the proposed framework.

II. METHODOLOGY

A. Behavior Trees

A behavior tree (BT) is a directed tree whose leaf nodes are called execution nodes and non-leaf nodes are called control flow nodes. It selects what should be done in the current environment and state through a tree-like decision structure similar to a decision tree. The execution of BTs starts from the root node, BTs continuously generates tick signals from the root node at a fixed frequency and propagates it to the children. When the node receives the tick signal, it starts to execute and returns the result S(Success), F(Failure) or R(Running) according to the execution situation, the execution result of the child node is controlled and managed by its parent node. A node in the R state will continue to execute until it returns S or F.

Nodes in BTs can be divided into two types: control flow nodes and execution nodes. There are three main control flow nodes: $Sequence$, $Fallback$ and $Parallel$, and two main types of execution nodes: $Action$ and $Condition$(Iovino et al., 2020).

Sequence: When $Sequence$ is running, it starts from the leftmost child node. When the child node returns S, the next child node starts to execute. When and only when all child nodes return S, $Sequence$ returns S.

Fallback: When $Fallback$ is running, it starts from the leftmost child node. If the child node returns F, the next child node starts to execute. As long as one child node returns S, $Fallback$ returns S. When all child nodes return F, $Fallback$ returns F.

Parallel: When $Parallel$ executes, all its child nodes begin to execute at the same time. Only when all the child nodes return S, $Parallel$ returns S. As long as one child node returns F, $Parallel$ returns F, and in other cases,

$Parallel$ returns R.

Action: This is the actual operation node of the robot, such as the movement, control or state change of the robot. Returning the actual running state of the robot.

Condition: It is a judgment node. Returning S or F according to whether the condition is true, and does not return R.

Common node types and their symbols of BT are shown in Table I.

TABLE I: The node types of behavior tree

Node type	Symbol	Success	Failure	Running
Sequence	$\rightarrow$	if all children return Success	if one child return Failure	if one child return Running
Selector	?	if one child return Success	if all children return Failure	if one child return Running
Parallel	||	if all children return Success	if one child return Failure	else
Action	$\square$	mission accomplished	if mission unable to complete	mission in progress
Condition	$\bigcirc$	conditions is true	conditions is false	never

B. Dynamic Movement Primitives

In this paper, DMPs is used to represent motor skills and used to encode motion trajectories. DMPs is essentially a second-order spring damping system, which can be divided into discrete and rhythmic types. In this work, we focus on the former. The DMPs can be expressed by the following formula (Ijspeert et al., 2001):

$$\tau \dot{v} = k(g - x) - dv + f(s) \tag{1}$$

$$\tau \dot{x} = v \tag{2}$$

$$\tau \dot{s} = -\alpha_1 s \tag{3}$$

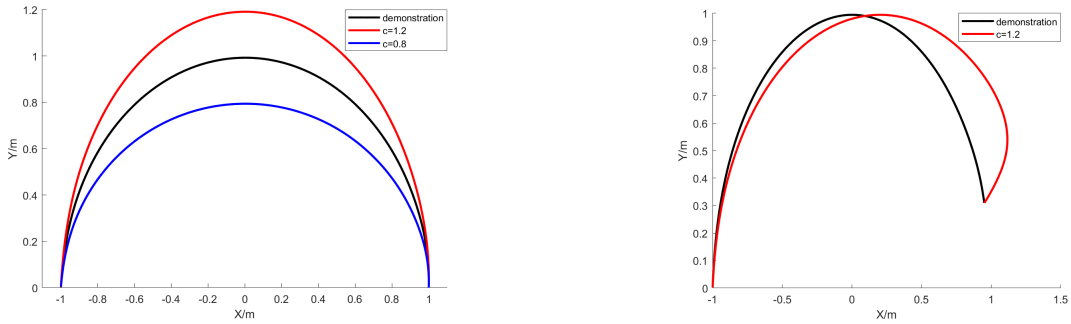
where Eq.(1) represents a transformation system consisting of a second-order spring damping system and a nonlinear function, x and v represent the position and velocity of the motion, respectively, k and d represent the spring constant and damping coefficient of the system, respectively, which are artificially designed parameters, usually let $k = d^2/4$, g denotes the target position of the motion, τ denotes the time scaling constant, s is the phase of the system, determined by Eq.(3). It decays from the initial value 1 to 0 with time, then model will become a stable second-order spring damping system. α_1 is a positive constant, $f(s)$ is a nonlinear function, which is defined as follows:

$$f(s) = \frac{\sum_{i=1}^{N} \psi_i \cdot \omega_i}{\sum_{i=1}^{N} \psi_i} \cdot (g - x_0)s \tag{4}$$

$$\psi_i = exp(-h_i(x - c_i)^2) \tag{5}$$

where c_i and h_i are the center and width of the i-th Gaussian function respectively, x_0 is the initial position, N is the number of Gaussian functions, and ω_i is the weight of the i-th Gaussian function.

The original DMPs allows translation and scaling of the trajectory, but the rotation in space causes distortion of its shape, because its nonlinearity in the three directions in space is learned separately and performed separately. In



(a) (b)

Fig. 1. (a)(b) is the trajectory scaled in the Z-axis direction. Scaling direction and direction vector are perpendicular in (a), but not in (b).

order to solve this problem, in (Koutras & Doulgeri, 2020), the author modified the original DMPs, which allows the trajectory to be arbitrarily rotated, translated, and scaled in space while keeping the shape unchanged. In (Si et al., 2021*a*), the author further proposed the following forms of DMPs:

$$\tau \dot{\boldsymbol{v}} = k(\boldsymbol{g} - \boldsymbol{x}) - d\boldsymbol{v} + s_g \boldsymbol{R_g} \boldsymbol{f(s)} \tag{6}$$

this is the representation of a three-dimensional trajectory, where $\boldsymbol{f(s)}$ is the same as the original DMPs, obtained by the demonstration trajectory, and s_g is the proportional coefficient, which can be obtained by the following formula:

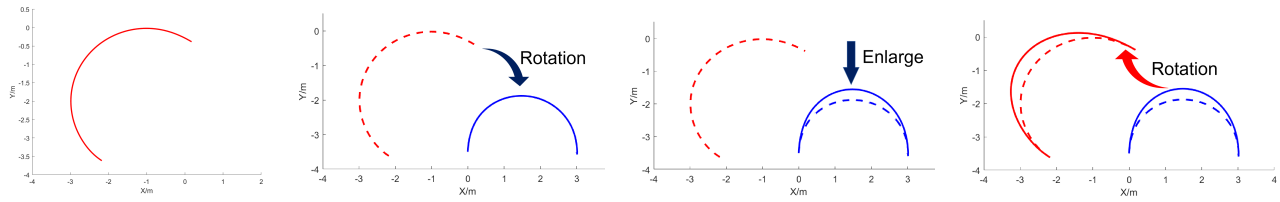
$$s_g = \frac{|| \boldsymbol{g} - \boldsymbol{x_0} ||}{|| \boldsymbol{g_d} - \boldsymbol{x_{0,d}} ||} \tag{7}$$

where $\boldsymbol{g}$ and $\boldsymbol{x_0}$ are the end point and start point of the demonstration trajectory, respectively, and $\boldsymbol{g_d}$ and $\boldsymbol{x_{0,d}}$ are the end point and start point of the recreation trajectory, respectively. $\boldsymbol{R_g}$ is the rotation matrix, can be obtained by the following formula:

$$\begin{cases} \boldsymbol{n} = \frac{\boldsymbol{g} - \boldsymbol{x_0}}{||\boldsymbol{g} - \boldsymbol{x_0}||}, \\ \boldsymbol{n_d} = \frac{\boldsymbol{g_d} - \boldsymbol{x_{0,d}}}{||\boldsymbol{g_d} - \boldsymbol{x_{0,d}}||}, \\ \hat{\boldsymbol{p}} = \frac{\boldsymbol{n} - \boldsymbol{n_d}}{||\boldsymbol{n} - \boldsymbol{n_d}||}, \\ \boldsymbol{S_k} = \left\{ \boldsymbol{k} \in \boldsymbol{R^3} | \boldsymbol{k^T} \hat{\boldsymbol{p}} = 0, ||\boldsymbol{k}|| = 1 \right\}, \end{cases} \tag{8}$$

$$\boldsymbol{R_g} = \boldsymbol{I_3} + \boldsymbol{S_k} \sin\theta + \boldsymbol{S_k^2}(1 - \cos\theta) \tag{9}$$

where $\boldsymbol{n}$ and $\boldsymbol{n_d}$ are the direction vector of demonstration and reproduction trajectory respectively, from the starting point to the end point of the trajectory. $\hat{\boldsymbol{p}}$ is dissecting plane, $\theta = \arccos(\boldsymbol{n_d^T n})$ is the minimum angle rotation between $\boldsymbol{n_d}$ and $\boldsymbol{n}$, $\boldsymbol{S_k}$ is the rotation axes. In order to further improve the applicability of DMPs, we have made further modifications to the above model. We believe that some trajectories with similar shapes can be considered the same skill. Taking the Pick and Place task as an example, robots need to cross different obstacles with different trajectories, which are actually only of varying amplitude in the vertical direction. Teaching and learning each



(a) (b) (c) (d)

Fig. 2. (a) is the original trajectory, (b) and (c) are the trajectory after rotation and scaling, respectively, (d) is to return the scaled trajectory to its original position.

trajectory is very inefficient. Therefore, We have made modifications to DMPs so that their shape can be scaled. The modified model is as follows:

$$\tau \dot{\boldsymbol{v}} = k(\boldsymbol{g} - \boldsymbol{x}) - d\boldsymbol{v} + \boldsymbol{L}\boldsymbol{f}(\boldsymbol{s}) \tag{10}$$

$$\boldsymbol{L} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \tag{11}$$

The above model realizes the scaling of the trajectory in the vertical direction, as shown in Fig. 1a, where a=b=1, and c is the scaling ratio of the trajectory in the vertical direction. However, the above model is only applicable when the direction vector of the track is perpendicular to the direction of scaling, that is, the scaling direction must be perpendicular to the line where the starting point and ending point of the track are located, otherwise, the trajectory will deform, as shown in Fig. 1b. For example, when the direction vector of the track is perpendicular to the Z-axis, it can be scaled in the Z-axis direction, when the direction vector is perpendicular to both the Z and Y-axis (parallel to the X-axis), it can be scaled in the corresponding two directions. And a, b, and c are the scaling ratios of the trajectory in the X, Y, and Z-axis directions, respectively. For the trajectory whose direction vector is not perpendicular to the coordinate axis, we can first rotate the track to meet the conditions, then scale it, and finally rotate it back to the original position, as shown in Fig. 2. The model is as follows:

$$\tau \dot{\boldsymbol{v}} = k(\boldsymbol{g} - \boldsymbol{x}) - d\boldsymbol{v} + \boldsymbol{R_0^{-1}} \boldsymbol{L} \boldsymbol{R_0} \boldsymbol{f}(\boldsymbol{s}) \tag{12}$$

where $\boldsymbol{R_0}$ is the rotation matrix that rotates the trajectory to a position that satisfies the condition, obtained according to Eq.(9), $\boldsymbol{L}$ is the scaling matrix, and $\boldsymbol{R_0^{-1}}$ is the rotation matrix that returns the trajectory to its original position. Trajectory can only be scaled in a direction perpendicular to the direction vector. We can select two fixed points that meet the conditions for rotation. For example, we can select two points on the X axis to determine $\boldsymbol{R_0}$. In this case, a=1, b and c are the scaling factor for height and width of the trajectory, $\boldsymbol{R_0^{-1}} \boldsymbol{L} \boldsymbol{R_0}$ is to be determined by b and c, denoted as $\boldsymbol{L}(b, c)$. In summary Eqs. (6)(7)(9)(12), the modified DMPs used in this article can be described as:

$$\tau \dot{\boldsymbol{v}} = k(\boldsymbol{g} - \boldsymbol{x}) - d\boldsymbol{v} + s_g \boldsymbol{L}(b, c) \boldsymbol{R_g} \boldsymbol{f}(\boldsymbol{s}) \tag{13}$$

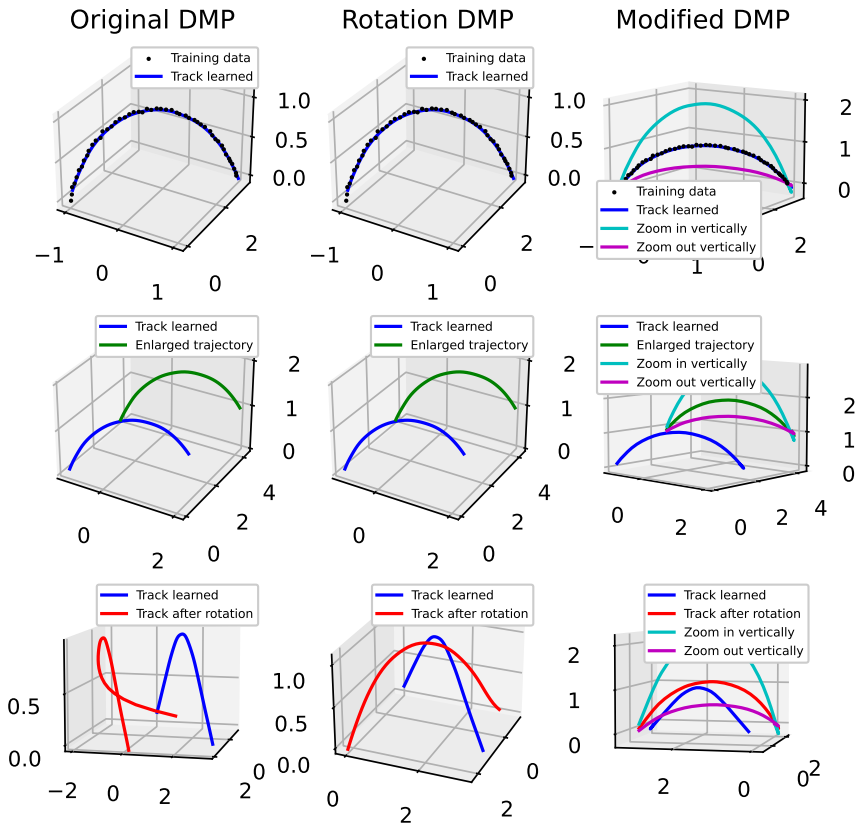


Fig. 3. The left column of the figures shows the original DMPs. It can be seen that the shape of the trajectory remains unchanged after translation, but deformation occurs after rotation. The middle column shows the Rotation DMPs, which can ensure that the trajectory does not deform after rotation. The right column shows the improved DMPs proposed in this paper, which is based on the rotated DMPs. It not only ensures that the trajectory does not deform after translation and rotation, but also allows for scaling, which further enhances its applicability.

Fig. 3 shows the comparison between the original DMPs, the DMPs in (Si et al., 2021*a*), and the modified DMPs proposed in this paper. It can be seen that the DMPs proposed in this article can translate, rotate, and zoom in and out of space after learning a basic skill, which enhances the applicability of the skill.

C. Broad Learning System

The BLS is an incremental learning method, which is improved and designed on the basis of random vector functional link neural network (RVFLNN) (Pao & Takefuji, 1992). The BLS does not directly input the original data into the network, but first performs feature mapping on the original input to obtain the feature layer, and then uses the feature layer to calculate the enhancement layer. The input of the network is the combination of the feature

layer and the enhancement layer. The schematic diagram of BLS is shown in Fig. 4. Where $\boldsymbol{X}$ and $\boldsymbol{Y}$ represent the input and output of the network, respectively, $\boldsymbol{Z^n} = [Z_1, ..., Z_n]$ and $\boldsymbol{H^m} = [H_1, ..., H_m]$ are the feature layer and the enhancement layer, respectively, obtained by the following formula:

$$Z_i = \phi(XW_{ki} + \beta_{ki}), i = 1, ..., n \tag{14}$$

$$H_i = \xi(ZW_{ej} + \beta_{ej}), j = 1, ..., m \tag{15}$$

where ϕ and ξ are transfer functions, W_{ki} and W_{ej} are randomly generated weight matrices, in order to prevent the correlation between the inputs of enhancement nodes, W_{ki} is generally orthogonal, and β_{ki} and β_{ej} are random deviations. The input of the network is $[Z^n|H^m]$, and the output is:

$$\begin{aligned} Y &= [Z_1, ..., Zn|\xi_2(Z^nW_{e1} + \beta_{e1}), ..., \xi_2(Z^nW_{em} + \beta_{em})]W \\ &= [Z_1, ..., Z_n|H_1, ..., H_m]W \\ &= [Z^n|H^m]W \end{aligned} \tag{16}$$

In order to improve the accuracy of the network, additional inputs can be added to the original network, and the addition of input data is the most important incremental form of BLS.

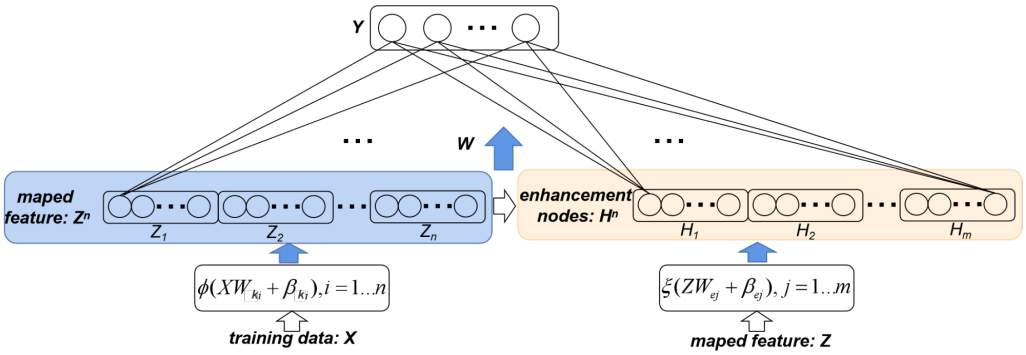


Fig. 4. Broad learning system. X is the original data. First, Z^n is obtained after certain transformation of X, then H^m is obtained after certain transformation of Z^n. Finally, Z^n and H^m are combined as the input of the network to obtain Y.

When there is new sample data, we use $A = [Z^n \mid H^m]$ to represent the initial input matrix of the mapping node and the enhancement node obtained from the initial data. Note that the new input sample is $[X_a \mid Y_a]$, and the input matrix of the mapping feature and the enhancement node obtained from X_a is A_x. The update formula is as follows:

$$D = A_x^T A^+ \tag{17}$$

$$C = A_x^T - DA \tag{18}$$

$$B = \begin{cases} C^+, & ifC \neq 0 \\ A^+D^T(I + DD^t)^{-1}, & ifC = 0 \end{cases} \tag{19}$$

The update formula of network weight is:

$$W^* = W + B(Y_a^T - A_x^T W) \tag{20}$$

The reason why we choose BLS is that compared with deep neural networks, BLS has the advantages of simple structure, fast training speed, easy expansion and online incremental learning.

D. Overview of the Proposed Method

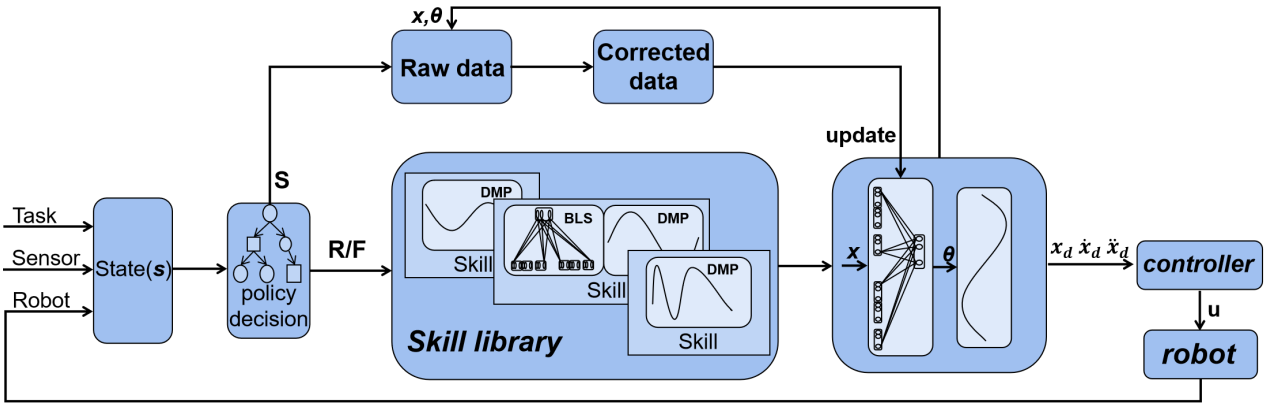


Fig. 5. Overview of the proposed method, BT selects the corresponding motor skills according to the system state s, S/R/F indicates three states of BT, BLS generates the motion parameters θ according to state x, which is part of s, and DMPs generates the control instructions to the controller. When the task is successfully completed, BLS will be updated.

We propose a framework that combines LfD, BT and BLS for robots to perform contact-rich manipulation tasks. The specific method is: for a complex task, first divide the task into multiple subtasks, get the motion primitives of each subtask by LfD, and express them with the modified DMPs. For some motion parameters in the task that cannot be determined in advance and need to be dynamically adjusted according to the environment, we use the BLS to generate these parameters according to the environment variables. Then build the BT of the task, the motion primitive and BLS act as nodes in the BT.

BT actually divides the robot and environment states into different state domains according to tasks, and specifies the behavior of the robot in different state domains. The purpose of each behavior is to let the robot go from the current state to the next expected state, so as to finally reach the target state. In a static known environment, the corresponding behavior in the state domain can be determined in advance, and the robot can successfully complete the task only by executing the motion command that is pre-planned by humans. However, in a dynamic or unknown environment, due to possible external interference, the robot may deviate from the expected state, or due to the imprecision of sensors, some external environments are unknown, which makes it difficult for the robot to move from the current state to the next expected state according to the preset motion. In order to solve the above problems, this paper proposes an iterative learning system based on BT and BLS.

Fig. 5 is the block diagram of our method, the input state of the system includes task parameters, data from external sensors and the state of the robot. BT will selects the next action based on the system state. We use DMPs to generate the trajectory of the robot. In these motions, the parameters of some motions can be preset or easily obtained, and they are usually motions that do not need to interact with the environment. However, for some motions that need to interact with the environment, their actions are difficult to determine in advance, and a strategy is needed to allow the robot to adjust in real time according to external stimuli during interaction with the environment. We use BLS to learn the strategy and generate DMPs parameters, in this paper, for some behaviors with uncertain parameters, we take the six-axis force information of the six-axis force sensor as the input of the BLS and output is the target position of the behavior. The BLS is generated by a small amount of training data, and its initial accuracy is not high, so it may not be able to make a correct strategy, that is, the robot cannot enter the next expected state from the current state domain, but may stay in the current state domain or enter another state domain. However, since the BT will traverse the entire tree from the root node at a fixed frequency, no matter whether there is external interference or the BLS makes an error response strategy, as long as the desired state has not been reached, the robot will always choose appropriate skills to move towards the target state, which can be considered as a process of resisting interference and trial and error. In this process, we record every input and output of BLS. If the robot finally completes the task, it is considered that the last strategy of the BLS is correct. We can correct the previous wrong output according to the last correct output, and update the weight of BLS based on the corrected data. The proposed method has the following three advantages:

1. The modified DMPs is used to characterize motion primitives, which enhances the generalization ability of tasks.

2. BT can enhance the anti-interference ability in the task execution.

3. BLS can be initialized based on a small number of samples. Because BT supports rollback, it can fall back to the previous state when BLS generates an incorrect strategy, and then update the weight value after success.

III. EXPERIMENT

We conducted experiments on cleaning tables and assembling peg-ring respectively.

A. Tabletop cleaning experiment

1) Task description: The experimental scene is shown in Fig. 6. The robot needs to clean the parts scattered on the table to one end of the table. The task is completed by three skills. Skill 1 and Skill 2 are pure displacement movements represented by DMPs. Skill 1 and 2 are used to reach the starting point of cleaning, return to the original track when encountering interference, and leave the table, and Skill 3 is composed of displacement movements and force control, which are used for wiping the desktop to ensure that the mechanical arm maintains a constant contact force with the desktop during the cleaning process. Because the desktop parts are scattered in a large range, it is impossible to clean all the parts at once, so the robot needs to clean the desktop repeatedly. The whole process will be completed by combining and generalizing the three skills through the BT model established according to the task. There are three parameters of the task, namely, the starting point, the end point and the direction of cleaning.

We carried out three experiments in total. In the first experiment, the robot needs to sweep the scattered parts from one end of the table to the other. In the second experiment, we added artificial interference on the basis of the previous experiment to make them deviate from the original cleaning track. In the third experiment, we changed the starting point and end point of cleaning, and rotated the cleaning direction by 90 degrees.

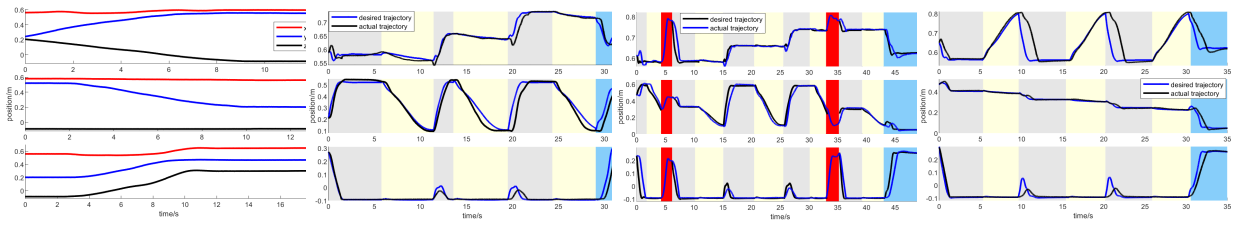
Fig. 6. The first and third lines show the robot com- pleting tasks from different directions, the blue arrow shows the sweeping direction, and the second line shows the robot returning to the sweeping position after encountering the disturbance and continuing to perform tasks.

2) Result Analysis: In the three experiments, the robot arm successfully completed the cleaning task, as follows:

(1) Experiment 1: The robot successfully completed three wiping tasks, and successfully swept the parts scattered on the table to one end of the table, as shown in Fig. 6. The process of robot moving to the next cleaning point after completing one cleaning is completed by skill 1. In order to ensure that the end-effector will not touch the parts on the desktop during this process, we let skill 1 scale twice in the vertical direction.

(2) Experiment 2: During wiping, we added two human disturbance. The disturbance was to drag the end-effector to make it deviate from the original trajectory, and the robot could successfully return to the original position and successfully completed the task, which shows that the proposed method has good anti-jamming performance. During this process, skill 1 and skill 2 were performed five times, and skill 3 was performed once. The motion trajectory of the end-effector is shown in Fig. 7c, where the red area represents the disturbance.

(3) Experiment 3: After changing the starting point and ending point, the whole sweeping path has been rotated 90 degrees, and the robot can still successfully complete the task, which fully demonstrates that the modified DMPs have good generalization performance. During this period, the motion trajectory of the end-effector is shown in



(a) (b) (c) (d)

Fig. 7. (a) is the three skills learned, (b)(c)(d) are respectively the trajectories of the end-effector in three experiments, the gray area is skill 1, the yellow area is skill 2, the blue area is skill 3, the red area represents human disturbance, The tasks in (b)(c)(d) are completed by generalization based on the three skills learned in (a).

Fig. 7d. It can be seen from the figure that the trajectory is quite different from that in Experiment 1.

The above three experiments show that using modified DMPs to model motor skills, and then using BT to combine different skills, can make tasks have better generalization ability and anti-jamming ability.

B. Peg-ring assembly experiment

1) Task description: We carried out a challenging peg-ring assembly experiment on the Elite mechanical arm. As shown in Fig. 8, a six-dimensional force sensor is installed at the end of the arm, and the peg is connected to the force sensor. This task requires inserting the peg into the ring and placing the ring on the desktop. Peg's diameter is 38mm, and the inner diameter of the ring is 40mm. The challenge of this task is that the arm needs to complete the insertion without knowing the exact position of the ring. At the beginning, the arm will try to insert from a wrong position. If the difference between the insertion position and the actual position of the hole center is more than 1mm, the peg will collide with the ring during the insertion process. We believe that the force is different when the insertion succeeds or fails, so we first judge whether the insertion succeeds according to the force. If the insertion fails, the force is different when the peg and ring contact at different positions, therefore, we can get the deviation between the current position and the accurate position of the ring according to the force during the insertion process, and then adjust the next insertion position according to this deviation. We set a threshold h_1, f is the force during insertion, if $|f| > h_1$, it is considered that the peg has contact with the environment, and then judge whether the peg is successfully inserted into the ring. We first collect the force f_d when the insertion is successful. If $|f - f_d| < h_2$, h_2 is also the set threshold, the insertion is successful, otherwise the insertion fails, and then predict the next insertion point based on f.

We use BLS as the strategy model, which is triggered by the contact force during the insertion process. The insertion position is adjusted according to the force detected during the insertion process. The input of the BLS is the six dimensional force information when the peg contacts the ring during the insertion process, and the output is a two-dimensional displacement bias, represents the deviation between the current position and the exact position

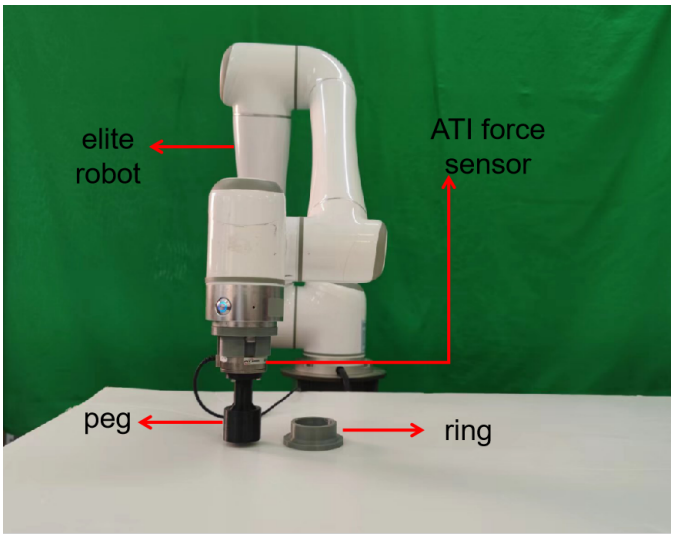


Fig. 8. Experiment scene diagram, Elite is a robot with six degrees of freedom, ATI is a six-dimensional force sensor, the diameter of peg is 38 mm, and the inner diameter of ring is 40 mm.

of the ring. The network is initialized by a small amount of training data, it is not convergent. The arm may need several attempts to insert the clevis into the ring, the input and output of BLS will be recorded at each insertion. After successful insertion, the recorded output data will be processed, and BLS will update the weight according to the processed data. In this way, after each successful completion of the task, the network will tend to converge more. Our goal is that after multiple insertions, the network can converge enough, that is, after only a few attempts, the peg can be inserted into the ring. The BT of the task is shown in Fig. 9.

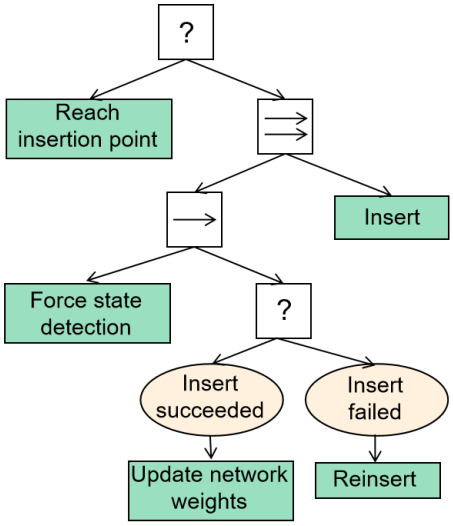


Fig. 9. Behavior tree of assembly task, this tree consists of a $Sequence$ node, two $Fallback$ nodes, a $Parallel$ node, two $Condition$ nodes, and five $Action$ nodes.

2) Initialization of network: The input of BLS is six-dimensional force information, and the output is the predicted ring center position. BLS is initialized by the collected training data, the training data is obtained by inserting at the position that is 2mm away from the actual position of the ring, as shown by the blue dot in Fig. 12.

3) Experimental process: First, we make the peg move randomly to a position within a certain range around the ring, and the peg is not aligned with the ring. The arm attempts to insert downward from the starting position. When the peg touches the ring, the BLS is triggered, it predicts the real position of the ring based on the current force information, and then the arm will re-inserted after aligning to the predicted position. If the BLS successfully predicts the exact position of the ring, the peg will be inserted successfully, otherwise the peg insertion will fail, the BLS be triggered again, and the arm will be aligned and inserted again until insert succeeded. Fig. 11 shows the successful insertion after 2 failures.

4) BLS weight update: Suppose that after n attempts, the peg is successfully inserted into the ring, and the n inputs and outputs of the neural network are $D = \{d_1, d_2, ..., d_n\}$, $d_i = \{f_i, x_i\}\,(1 \leq i \leq n)$. f_i is the contact force when the insertion fails, and x_i is the deviation between the current position of the network output and the accurate position of the ring. We believe that only the last outputs of BLS is correct, and the previous $n-1$ times are wrong. After the peg is finally inserted into the ring successfully, we assume that the final insertion position p^* is the exact position of the ring. Therefore, the correct output of BLS should be the deviation from the insertion position to p^*. The actual deviation corresponding to the contact force f_i during the i_{th} insertion shall be

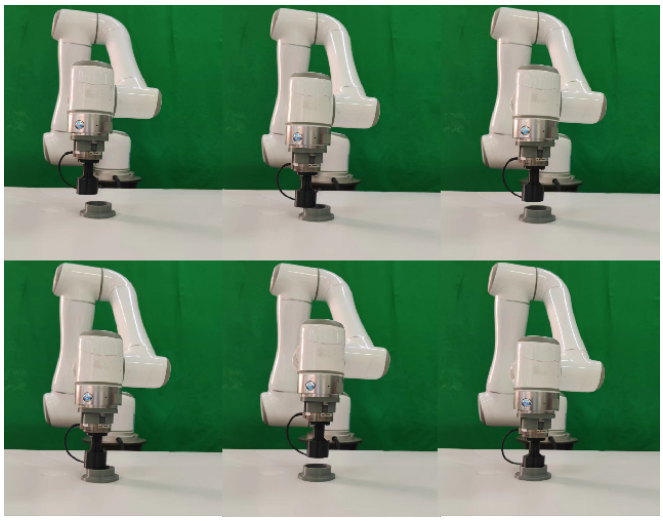


Fig. 10. Insertion completed successfully after 2 failures, First, the arm tries to insert downward at the initial point. After contacting the ring, BLS predicts the next insertion position according to the force information. The arm inserts downward again, but still fails. BLS predicts the next insertion position again, and finally inserts successfully.

$x_i^* = \sum_i^n x_i\ (1 \leq i \leq n-1)$. Fig. 11 shows an example of successfully completing the task after three attempts. Each attempt will be used as a new sample to update the weight. After correcting the wrong output, update the

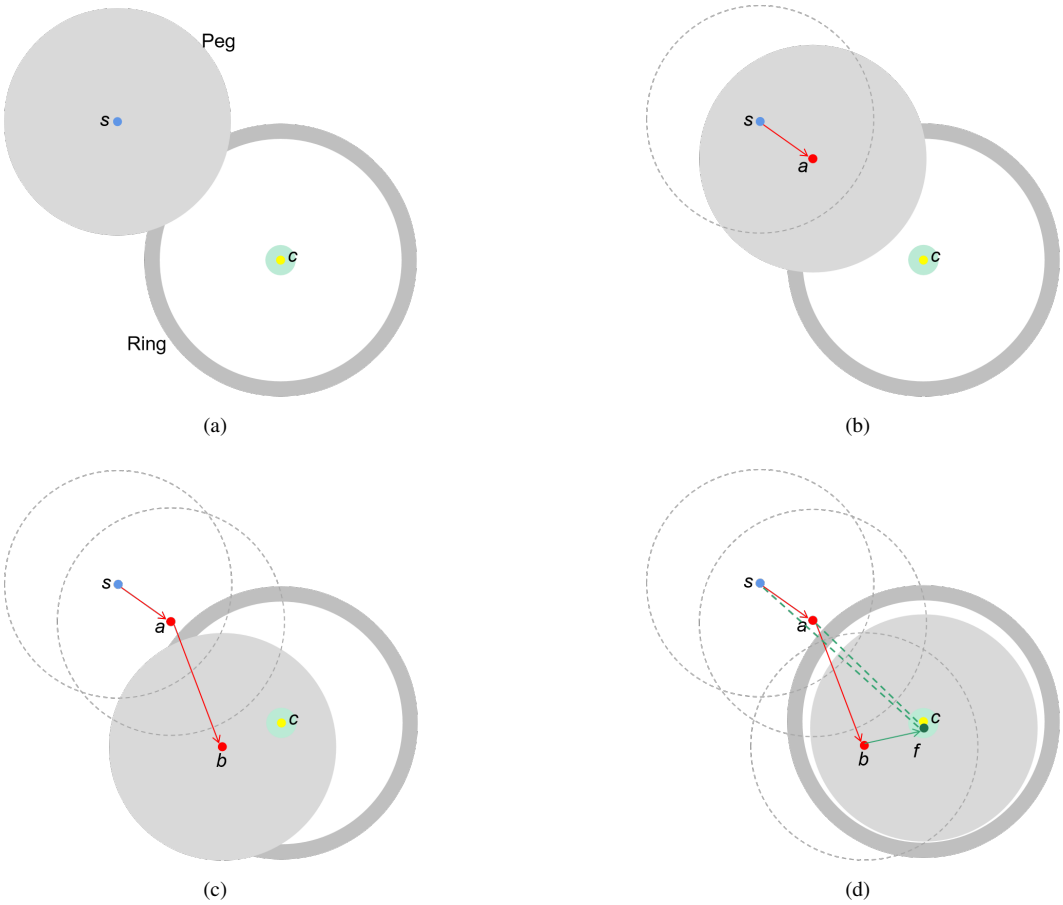


Fig. 11. The peg starts from s, and after two attempts at a and b, it is finally inserted successfully at f. c is the midpoint of the ring, and the green area indicates the area that can be successfully inserted. The collision force and $\overrightarrow{sf}$, $\overrightarrow{af}$, $\overrightarrow{bf}$ of the peg at s, a, b insertion will be used as the output and input of new samples for updating weights.

weight of BLS based on Eq. (17)-(20).

5) Result analysis: We carried out a total of 100 insertion tasks, and the initial positions of each insertion were randomly obtained from the range of 1-5mm from the center of the ring. The initial insertion positions of these 100 times are shown in Fig. 12. Table II shows the average number of attempts for successful insertion. It can be seen that at the beginning, because the BLS is not convergent, the arm needs 11.7 attempts on average to successfully insert. However, with the increase of the number of successful attempts, the weight of the BLS tends to converge. After 20 experiments, the insertion can be successfully completed by 3.4 attempts on average. However, as the training data continues to increase, it cannot achieve the effect of successful insertion only once. We think there are two reasons: 1. The measurement error of the force sensor. 2. The expected network output is the deviation from the insertion position to the center of the ring. However, since the position of each successful insertion is not

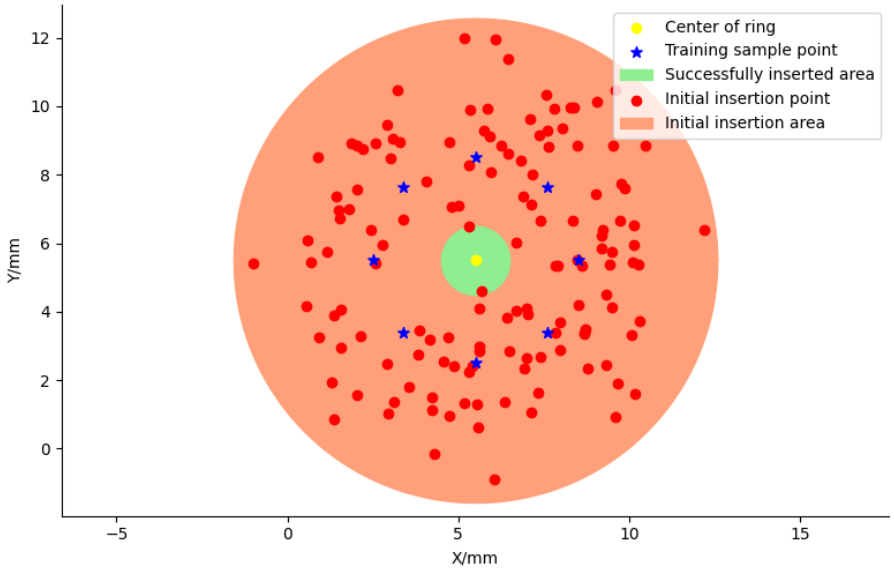


Fig. 12. The blue point and the red point are the sample points and the random initial points in the experiment, respectively. The green area is the area that can successfully complete the insertion task.

necessarily the center of the ring, the correction data used to update the network is not completely correct.

TABLE II: Average number of successful insert attempts

Number of experiments	0	5	10	20	50	100
Average Attempts	11.7	8.3	5.4	3.4	2.3	2.1

IV. Conclusion

We proposed a human-robot skill transfer framework based on modified DMPs and BT. Due to the simplicity of the DMPs and its strong generalisation ability, it was adopted to encode manipulation skills, clean tables and assemble. Furthermore, the proposed framework enables robots to recover from human interference during the execution of a skill. In addition, BLS is integrated into the framework and its weight is iteratively optimized, which further enhances the generalization and applicability of the framework. In this work, BLS was used to generate specific parameters of motion. In future work, we will further refine and improve the proposed method, such as using BLS to select motor skills, so that it could complete more complex manipulation tasks and have better generalisation capability.

References

Bohren, J. & Cousins, S. (2010), 'The smach high-level executive [ros news]', *IEEE Robotics & Automation Magazine* **17**(4), 18–20.

Calinon, S. & Billard, A. (2009), 'Statistical learning by imitation of competing constraints in joint space and task space', *Advanced Robotics* **23**(15), 2059–2076.

Han, L., Yuan, H., Xu, W. & Huang, Y. (2022), 'Modified dynamic movement primitives: Robot trajectory planning and force control under curved surface constraints', *IEEE Transactions on Cybernetics* .

Hauge, T. C., Katz, G. E., Davis, G. P., Huang, D.-W., Reggia, J. A. & Gentili, R. J. (2021), 'High-level motor planning assessment during performance of complex action sequences in humans and a humanoid robot', *International Journal of Social Robotics* **13**(5), 981–998.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P. & Schaal, S. (2013), 'Dynamical movement primitives: learning attractor models for motor behaviors', *Neural computation* **25**(2), 328–373.

Ijspeert, A. J., Nakanishi, J. & Stefan, S. (2001), Trajectory formation for imitation with nonlinear dynamical systems, *in* 'Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)', Vol. 2, IEEE, pp. 752–757.

Iovino, M., Scukins, E., Styrud, J., Ögren, P. & Smith, C. (2020), 'A survey of behavior trees in robotics and ai', *arXiv preprint arXiv:2005.05842* .

Koutras, L. & Doulgeri, Z. (2020), A novel dmp formulation for global and frame independent spatial scaling in the task space, *in* '2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)', IEEE, pp. 727–732.

Kulvicius, T., Ning, K., Tamosiunaite, M. & Worgötter, F. (2011), 'Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting', *IEEE Transactions on Robotics* **28**(1), 145–157.

Liao, B. & Liu, W. (2015), 'Pseudoinverse-type bi-criteria minimization scheme for redundancy resolution of robot manipulators', *Robotica* **33**(10), 2100–2113.

Liao, B., Wang, Y., Li, W., Peng, C. & Xiang, Q. (2021), 'Prescribed-time convergent and noise-tolerant z-type neural dynamics for calculating time-dependent quadratic programming', *Neural Computing and Applications* **33**, 5327–5337.

Lončarević, Z., Pahič, R., Ude, A., Gams, A. et al. (2021), 'Generalization-based acquisition of training data for motor primitive learning by neural networks', *Applied Sciences* **11**(3), 1013.

Pao, Y.-H. & Takefuji, Y. (1992), 'Functional-link net computing: theory, system architecture, and functionalities', *Computer* **25**(5), 76–79.

Pastor, P., Hoffmann, H., Asfour, T. & Schaal, S. (2009), Learning and generalization of motor skills by learning from demonstration, *in* '2009 IEEE International Conference on Robotics and Automation', IEEE, pp. 763–768.

Pignat, E., Silvério, J. & Calinon, S. (2022), 'Learning from demonstration using products of experts: Applications to manipulation and task prioritization', *The International Journal of Robotics Research* **41**(2), 163–188.

Prada, M., Remazeilles, A., Koene, A. & Endo, S. (2013), Dynamic movement primitives for human-robot interaction: comparison with human behavioral observation, *in* '2013 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 1168–1175.

Qiao, H., Wu, Y.-X., Zhong, S.-L., Yin, P.-J. & Chen, J.-H. (2023), 'Brain-inspired intelligent robotics: Theoretical

analysis and systematic application', *Machine Intelligence Research* **20**(1), 1–18.

Qiao, H., Zhong, S., Chen, Z. & Wang, H. (2022), 'Improving performance of robots using human-inspired approaches: a survey', *Science China Information Sciences* **65**(12), 1–31.

Ravichandar, H., Polydoros, A. S., Chernova, S. & Billard, A. (2020), 'Recent advances in robot learning from demonstration', *Annual review of control, robotics, and autonomous systems* **3**, 297–330.

Rozo, L., Jiménez, P. & Torras, C. (2011), Robot learning from demonstration of force-based tasks with multiple solution trajectories, *in* '2011 15th International Conference on Advanced Robotics (ICAR)', IEEE, pp. 124–129.

Si, W., Wang, N. & Yang, C. (2021*a*), Reactive and constrained motion primitive merging and adaptation, *in* '2021 26th International Conference on Automation and Computing (ICAC)', IEEE, pp. 1–6.

Si, W., Wang, N. & Yang, C. (2021*b*), 'A review on manipulation skill acquisition through teleoperation-based learning from demonstration', *Cognitive Computation and Systems* **3**(1), 1–16.

Si, W., Yue, T., Guan, Y., Wang, N. & Yang, C. (2022), A novel robot skill learning framework based on bilateral teleoperation, *in* '2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)', IEEE, pp. 758–763.

Styrud, J., Iovino, M., Norrlöf, M., Björkman, M. & Smith, C. (2022), Combining planning and learning of behavior trees for robotic assembly, *in* '2022 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 11511–11517.

Wang, R., Wu, Y., Chan, W. L. & Tee, K. P. (2016), 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)', IEEE, pp. 3765–3771.

Wu, Y., Wang, R., D'Haro, L. F., Banchs, R. E. & Tee, K. P. (2018), 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)', IEEE, pp. 1–7.

Yang, C., Zeng, C., Fang, C., He, W. & Li, Z. (2018), 'A dmps-based framework for robot learning and generalization of humanlike variable impedance skills', *IEEE/ASME Transactions on Mechatronics* **23**(3), 1193–1203.

Zhang, Y., Li, W., Liao, B., Guo, D. & Peng, C. (2014), 'Analysis and verification of repetitive motion planning and feedback control for omnidirectional mobile manipulator robotic systems', *Journal of intelligent & robotic systems* **75**, 393–411.