Computers & Industrial Engineering 191 (2024) 110144

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Probabilistic movement primitives based multi-task learning framework

Chengfei Yue [a], Tian Gao [a,*], Lang Lu [a], Tao Lin [b], Yunhua Wu [c]

[a] Institute of Space Science and Applied Technology, Harbin Institute of Technology, Shenzhen 518055, China
[b] Research Center of the Satellite Technology, Harbin Institute of Technology, Harbin 150001, China
[c] School of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China

ARTICLE INFO

Keywords:
Learning from Demonstration
Conditional Probabilistic Movement Primitives
Learning beyond teaching
Multi-task learning

ABSTRACT

With the increasing complexity of industrial production and manufacturing tasks, industrial robots are expected to learn intricate operations from simple actions easily and quickly with adaption to dynamic environment. In this paper, a task-parameterized multi-task learning framework is proposed to facilitate rapid learning of operational skills for industrial robots. In this framework, a conditional Probabilistic Movement Primitives (ProMP) is firstly employed to the single-task learning. Using the conditional probability calculation, the extrapolation issue in Learning from Demonstration (LfD) is addressed, enabling robots to learn beyond teaching. Subsequently, the single-task is extended to multi-task scenario by proposing a multi-task learning approach where each single task executes an extrapolation learning. The learned skill can meet the multiple task requirements through an iterative modulation manner. The effectiveness of the proposed framework is validated through both the simulation and a 7-DoF Franka-Emika robot experiment in a predefined task scenario. Furthermore, the outperformance of the proposed method is demonstrated by comparing with the state-of-art movement primitives based learning method.

1. Introduction

The rapid advancement of industrial automation technology has made industrial robots increasingly important in production and manufacturing processes. The traditional robot task execution relies on pre-programming method, which requires programming expertise and significant time investment (Ravichandar, Polydoros, Chernova, & Billard, 2020). Moreover, the programming has to be repeated for different tasks, lacking flexibility in handling dynamic tasks. To better adapt to complex production environments and tasks, more and more industrial robots are equipped with intelligent systems that learn skills from a series of simple actions implicitly to replace the explicit manual programming (Ying, Pourhejazy, Cheng, & Cai, 2021). Learning from Demonstration (LfD), which involves imitating an expert to gain similar skills with more flexibility to handle the constraints and requirements, has become a promising branch of industrial robot intelligence (Argall, Chernova, Veloso, & Browning, 2009; Schaal, 1996).

The LfD process involves recording a number of movements during a task, extracting motion features from demonstration samples, and subsequently generalizing these features to new scenarios. According to different demonstration technologies, LfD approaches can be classified into three categories: kinesthetic teaching (Elliott, Xu, & Cakmak, 2017), teleoperation (Park, Ntuen, Vootla, & Park, 1994) and passive observation (Ravichandar et al., 2020). Kinesthetic teaching enables skill learning through physical interaction between humans and robots, eliminating additional interaction devices and simplifying the learning process (Kormushev, Calinon, & Caldwell, 2011). Teleoperation allows remote teaching, but it introduces additional teaching input interfaces and lacks user-friendliness for inexperienced demonstrators. In contrast, passive observation transfers the complexity from the demonstrator to the learner, where the mapping from human's behaviors to robot's actions remains quite complex. Considering the pros and cons of these technologies, the kinesthetic teaching becomes attractive in this research due to the readily available experimental data, such as joint state (Gomez-Gonzalez, Neumann, Schölkopf, & Peters, 2020), end-effector state (Khansari-Zadeh & Billard, 2011; Rana, Mukadam, Ahmadzadeh, Chernova, & Boots, 2018) and end-effector force (Calinon, Evrard, Gribovskaya, Billard, & Kheddar, 2009; Kober, Gienger, & Steil, 2015).

In the field of learning demonstration data, the probability based approach, with its excellent data modeling and representation capability, has been widely used (Allen, Roychowdhury, & Liu, 2018; Peng et al., 2021). Lin, Ren, Clevenger, and Sun (2012) modeled the force and position trajectory of the robot manipulator's fingertips with the Gaussian Mixture Model (GMM) and generated the reproduced trajectory via Gaussian Mixture Regression (GMR), achieving the effective

* Corresponding author.
E-mail address: 22S061022@stu.hit.edu.cn (T. Gao).

https://doi.org/10.1016/j.cie.2024.110144
Received 12 October 2023; Received in revised form 20 March 2024; Accepted 8 April 2024
Available online 10 April 2024

learning of the demonstrated skills. The limitation of GMM is its inability to automatically determine the appropriate number of GMM components. To address this issue, a Dirichlet GMR model with a nonparametric Bayesian formulation is introduced in Chatzis, Korkinof, and Demiris (2012), which implements automatic clustering. Besides, although GMM is capable of learning probabilistic features from multiple samples, it lacks the generalization capability, which refers to applying the learned skills from the demonstration environment to different scenarios. Moreover, when learning high-dimensional data, the GMM model becomes more complex.

To overcome these limitations, researchers are motivated to develop the Movement Primitives (MP) that can model high-dimensional data easily and generalize trajectory promptly (Flash & Hochner, 2005; Mussa-Ivaldi, 1999). MP combines and recombines a series of basic motion patterns to generate more complex movements by modifying their relationships. Rozo, Calinon, Caldwell, Jiménez, and Torras (2013) and Rozo, Calinon, Caldwell, Jimenez, and Torras (2016) employed a heuristic-based approach using multiple motion primitives to encode and reproduce impedance behaviors of a task-parameterized statistical dynamical system. Schaal (2006) proposed Dynamic Movement Primitives (DMP) as a typical representative of MP. DMP utilizes a second-order spring-damper system with self-stability to construct an "attractor" model. Changing the "attractor point" with a nonlinear forcing term can alter the system's target, thereby modulating the trajectory's final state point where the reproduced trajectory has to pass through (Ijspeert, Nakanishi, Hoffmann, Pastor, & Schaal, 2013; Saveriano, Abu-Dakka, Kramberger, & Peternel, 2021; Ude, Gams, Asfour, & Morimoto, 2010). However, the DMP-based approaches converge to a final zero velocity, which significantly astricts the implementation of these methods. Furthermore, the learning mechanism of DMP implies that it is incapable of modulating via-points' state of trajectory. Thus, for specific tasks where some via-points' constraints and non-zero final velocity states are required, the DMP is invalid. To handle the trajectory modulation problem, Probabilistic Movement Primitives (ProMP) is implemented. In ProMP, the demonstration trajectories are assumed to represent samples of an underlying stochastic process. The learned probability distribution is modulated to derive a new distribution, from which the trajectory that satisfies state constraints at arbitrary points, including positions and velocities at the initial, intermediate and final points on the trajectory, is generated (Maeda et al., 2017; Paraschos, Daniel, Peters, & Neumann, 2013, 2018).

These mentioned probabilistic methods perform good generalization inside the area covered by the demonstration. However, they tend to generate unpredictable motions when the required actions deviate from the demonstration region significantly, defined here as the extrapolation problem in LfD. Therefore, one major challenge of extrapolation is to generalize outside the teaching while reflecting and maintaining the characteristics of the initial demonstration, in other words, preserving the overall shape of the initial demonstration. In order to address this issue, one feasible way of these methods requires a larger number of demonstration trajectories, which increases the burden of LfD. To prevent trajectory deformation during extrapolation, Burlizzi, Vochten, De Schutter, and Aertbeliën (2022) utilized the invariants method to ensure the shape preservation of trajectories. Zhou, Gao, and Asfour (0000) introduced a Via-points Movement Primitives, which enables extrapolation of arbitrary via-points on the trajectory. Another researches aim to combine LfD with Reinforcement Learning (RL) where RL is used to improve and extend the learning performance of LfD, such as the policy gradient (Akbulut et al., 2021) and neural network (Perez-Villeda, Piater, & Saveriano, 2023). However, the iterative process in RL is usually time-consuming and cannot fulfill the real-time requirement of the task. Besides, these limited researches primarily focus on extrapolation of the trajectory, whereas simple trajectory planning alone falls short in the complex multi-task scenarios.

In this paper, aiming at handling the extrapolation in single-task scenario and fast learning of multiple tasks, a task-parameterized multi-task learning framework is presented. This framework utilizes a conditional ProMP to facilitate robots' rapid learning for the single task by replacing iteration with computation. Additionally, the use of conditional probability calculation addresses the extrapolation issue in LfD, enabling robots to learn beyond teaching. Furthermore, the extrapolation learning method of single task is extended to handle the multi-task problem and a multi-task rapid learning approach is developed to complete multiple tasks simultaneously. The contribution of this paper lies in the following two aspects: (1) the extrapolation problem for a single task is addressed which enables the robot learn beyond teaching; (2) a rapid learning via computation rather than iterative learning is utilized to fulfill the multi-task requirements.

This paper is organized as follows: Section 2 introduces the preliminaries of ProMP. Building upon this foundation, the proposed learning framework is presented in Section 3. In Section 4, the effectiveness and performance of the proposed framework are demonstrated through the simulation and experiment using a 7-DOF Franka-Emika robot as well as the comparison with the movement primitives based learning method. Finally, the conclusive remarks and future work prospects are made in Section 5.

2. Preliminaries of ProMP

2.1. ProMP representation and reproduction

ProMP essentially represents a probability distribution over the motion trajectories. Considering a robot system with a single Degree-of-Freedom (DoF) joint for simplicity, and a joint trajectory is denoted by $\boldsymbol{\xi} = \{\boldsymbol{y}_t\}_{t=1}^{T}$, which is a joint state sequence sampled from demonstration with a length T. Here, $\boldsymbol{y}_t \in \mathbb{R}^2$ includes the joint angular position and velocity at time step $t \in [1, 2, \ldots, T]$. With the application of Gaussian Linear Regression (GLR), the trajectory can be approximated by a linear combination of Gaussian basis functions, yielding:

$$\boldsymbol{y}_t = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \boldsymbol{\Phi}_t \boldsymbol{\omega} + \boldsymbol{\varepsilon}_y \tag{1}$$

where $\boldsymbol{\Phi}_t = \begin{bmatrix} \boldsymbol{\phi}_t & \dot{\boldsymbol{\phi}}_t \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{2\times N}$ is the matrix of basis functions at time step t, N is the number of basis functions and $\boldsymbol{\omega} \in \mathbb{R}^N$ denotes the weight vector. To account for the potential error between the approximation and actual data, the Gaussian noise is introduced with the form of $\boldsymbol{\varepsilon}_y \sim \mathcal{N}\left(0, \boldsymbol{\Sigma}_y\right) \in \mathbb{R}^N$.

According to Eq. (1), the probability distribution of joint state at time step t given the weight $\boldsymbol{\omega}$ condition can be written as:

$$p\left(\boldsymbol{y}_t \,|\boldsymbol{\omega}\right) = \mathcal{N}\left(\boldsymbol{y}_t \,\middle|\boldsymbol{\Phi}_t \boldsymbol{\omega}, \boldsymbol{\Sigma}_y\right) \tag{2}$$

Suppose that the robot is taught kinesthetically for a certain task with M times, resulting a dataset containing M weight vectors denoted as $\boldsymbol{W} = \{\boldsymbol{\omega}^m\}_{m=1}^{M}$. In order to better reflect the generality and diversity of the weights, they can be modeled as a multi-dimensional Gaussian distribution:

$$p\left(\boldsymbol{\omega}; \boldsymbol{\theta}_{\boldsymbol{\omega}}\right) = \mathcal{N}\left(\boldsymbol{\omega} \,|\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}}\right) \tag{3}$$

where $\boldsymbol{\mu}_{\boldsymbol{\omega}} \in \mathbb{R}^N$ and $\boldsymbol{\Sigma}_{\boldsymbol{\omega}} \in \mathbb{R}^{N\times N}$ respectively represent the mean vector and covariance matrix of the weight distribution. The distribution parameters can be learned from $\boldsymbol{W}$ using the principle of Maximum Likelihood Estimation (MLE).

Therefore, the distribution of the joint state at time step t can be easily derived by combining Eqs. (2) and (3) as follows:

$$\begin{aligned} p\left(\boldsymbol{y}_t; \boldsymbol{\theta}_{\boldsymbol{\omega}}\right) &= \int p\left(\boldsymbol{y}_t \,|\boldsymbol{\omega}\right) p\left(\boldsymbol{\omega}; \boldsymbol{\theta}_{\boldsymbol{\omega}}\right) \mathrm{d}\boldsymbol{\omega} \\ &= \int \mathcal{N}\left(\boldsymbol{y}_t \,\middle|\boldsymbol{\Phi}_t \boldsymbol{\omega}, \boldsymbol{\Sigma}_{\mathrm{y}}\right) \mathcal{N}\left(\boldsymbol{\omega} \,|\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}}\right) \mathrm{d}\boldsymbol{\omega} \\ &= \mathcal{N}\left(\boldsymbol{y}_t \,\middle|\boldsymbol{\Phi}_t \boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{\omega}} \boldsymbol{\Phi}_t^{\mathrm{T}} + \boldsymbol{\Sigma}_y\right) \end{aligned} \tag{4}$$

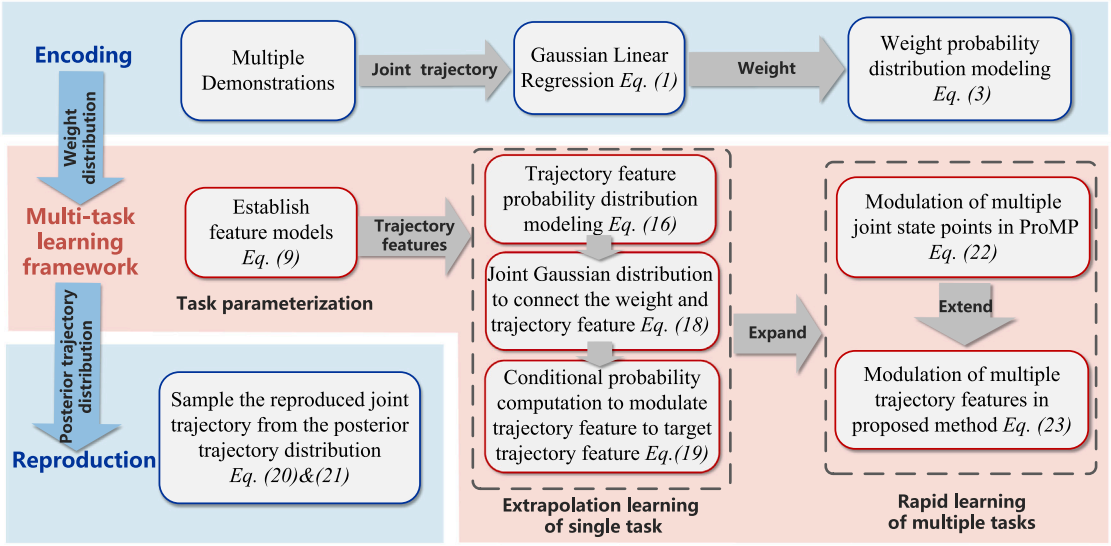

Fig. 1. The overall learning process.

Furthermore, according to the principle of probability superposition, the distribution of joint trajectory can be expressed as:

$$p\left(\xi;\theta_{\omega}\right)=\prod_{T}p\left(\boldsymbol{y}_{t};\theta_{\omega}\right) \tag{5}$$

Then, if there is no additional requirement, a trajectory with the highest sampling probability described by the distribution can be reproduced.

2.2. ProMP generalization

A fundamental advantage of ProMP is the trajectory generalization, which refers to modulate the joint state (i.e., angular position and velocity) of the given time step. This modulation ensures that the joint trajectory smoothly passes through the desired joint state point at specific time step. Specifically, assuming the robot is required to pass through an expected observation point $\boldsymbol{x}_{t}^{*}=\{\boldsymbol{y}_{t}^{*},\boldsymbol{\Sigma}_{y}^{*}\}$ at a certain time step t^{*}, where the desired joint state are represented by $\boldsymbol{y}_{t}^{*}$, with $\boldsymbol{\Sigma}_{y}^{*}$ denoting the allowable error noise between the actual and desired joint states. In probabilistic model, the expected observation point can be incorporated through conditioning and applying Bayes' theorem (O'Hogan & Foster, 1999), it is obtained that:

$$p\left(\boldsymbol{\omega}\,|\boldsymbol{x}_{t}^{*}\right)\propto\mathcal{N}\left(\boldsymbol{y}_{t}^{*}\,|\boldsymbol{\Phi}_{t^{*}}\boldsymbol{\omega}\,,\boldsymbol{\Sigma}_{y}^{*}\right)p\left(\boldsymbol{\omega}\right) \tag{6}$$

Accordingly, the weight distribution is modified to a new one and the new mean and covariance parameters can be calculated through conditioning using the following formula:

$$\begin{cases}\boldsymbol{\mu}_{\omega}^{[\mathrm{new}]}=\boldsymbol{\mu}_{\omega}+\boldsymbol{\Sigma}_{\omega}\boldsymbol{\Phi}_{t^{*}}^{\mathrm{T}}\left(\boldsymbol{\Sigma}_{y}^{*}+\boldsymbol{\Phi}_{t^{*}}\boldsymbol{\Sigma}_{\omega}\boldsymbol{\Phi}_{t^{*}}^{\mathrm{T}}\right)^{-1}\left(\boldsymbol{y}_{t}^{*}-\boldsymbol{\Phi}_{t^{*}}\boldsymbol{\mu}_{\omega}\right)\\ \boldsymbol{\Sigma}_{\omega}^{[\mathrm{new}]}=\boldsymbol{\Sigma}_{\omega}-\boldsymbol{\Sigma}_{\omega}\boldsymbol{\Phi}_{t^{*}}^{\mathrm{T}}\left(\boldsymbol{\Sigma}_{y}^{*}+\boldsymbol{\Phi}_{t^{*}}\boldsymbol{\Sigma}_{\omega}\boldsymbol{\Phi}_{t^{*}}^{\mathrm{T}}\right)^{-1}\boldsymbol{\Phi}_{t^{*}}\boldsymbol{\Sigma}_{\omega}\end{cases} \tag{7}$$

where $\boldsymbol{\mu}_{\omega}^{[\mathrm{new}]},\boldsymbol{\Sigma}_{\omega}^{[\mathrm{new}]}$ are the new mean and covariance of the weight distribution respectively. After modifying the weight distribution, the satisfactory trajectory can be obtain using Eqs. (4) and (5), where the desired joint state point is passed through.

Furthermore, when extending ProMP to the multi-DoF scenario with a number of DoF n, Eq. (1) can be reformulated as:

$$p\left(\boldsymbol{Y}_{t}\,|\boldsymbol{\Omega}\right)=\mathcal{N}\left(\begin{bmatrix}\boldsymbol{y}_{1,t}\\ \vdots\\ \boldsymbol{y}_{n,t}\end{bmatrix}\middle|\begin{bmatrix}\boldsymbol{\Phi}_{t} & \cdots & \boldsymbol{0}\\ \vdots & \ddots & \vdots\\ \boldsymbol{0} & \cdots & \boldsymbol{\Phi}_{t}\end{bmatrix}\begin{bmatrix}\boldsymbol{\omega}_{1}\\ \vdots\\ \boldsymbol{\omega}_{n}\end{bmatrix},\boldsymbol{\Sigma}_{Y}\right) \tag{8}$$

where $\boldsymbol{y}_{i,t}$ represents the state vector for the ith joint at time step t.

More details about ProMP can be found in Paraschos et al. (2013, 2018).

3. Task-parameterized multi-task rapid learning framework

The overall learning process in this paper is shown in Fig. 1, including: encoding, learning and reproduction. The encoding and reproduction follow the traditional ProMP approach, as described in Section 2. Then in the learning step, a multi-task learning framework is proposed, which includes the task parameterization (Section 3.1), extrapolation learning of single task (Section 3.2) and rapid learning of multiple tasks (Section 3.3). The concepts of trajectory feature (i.e., Eq. (9)) and target trajectory feature are firstly introduced in the task parameterization phase. Specifically, the completeness of a trajectory for a task is characterized by its trajectory feature, and when the feature meets the target trajectory feature, the task can be completed. In the extrapolation learning of single task phase, a joint Gaussian distribution connecting the weight and trajectory feature is established. By employing the conditional probability calculation, the trajectory feature is modulated to the target trajectory feature to satisfy the task demand following Eqs. (16) to (19). Furthermore, the single-task is expanded to multi-task scenario in the multi-task learning phase. The modulation of multiple joint state points in ProMP is extended to the modulation of multiple trajectory features in the proposed method. Multiple tasks are iteratively processed with each task performing an extrapolation learning, where a trajectory feature is modulated to a target trajectory feature for a single task, ultimately all trajectory features are modulated to their corresponding target features. Thus, the reproduced trajectory can complete multi-task learning.

3.1. Task parameterization

In essence, the goal of skill learning in robotics is to discover a trajectory that can meet the task requirement, either in joint space or in end-effector Cartesian space (In this paper, we focus on trajectory learning in joint space). Therefore, before we move forward to introduce the newly developed learning framework, some new insights about the trajectory and task should be prepared.

Trajectory feature: a trajectory related real value, which quantifies the degree to which the trajectory contributes to a specific task completion. For example, in a grasping task, the distance between the gripper and the target can be considered as a trajectory feature to evaluate the degree to complete the grasping task.

Here, the general feature function to compute the trajectory feature is given as follows:

$$\boldsymbol{C}_{i}\left(\boldsymbol{y}\right)=\int_{t=1}^{T}f_{i}\left(\boldsymbol{y}_{t}\right)\mathrm{d}t \tag{9}$$

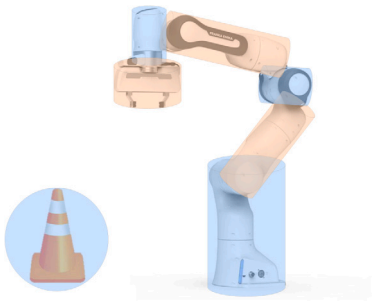

Fig. 2. Boolean element body envelopment illustration.

with $f_i(\boldsymbol{y}_t)$ being a feature function of the joint state in the ith task.

Target trajectory feature: a specific real value in the domain of trajectory feature and relates to a specific task. It is referred to the target value of the aforementioned trajectory feature to meet the task requirement or complete the task. When the target trajectory feature is met, the task can be completed. For the same grasping task, the distance between the gripper and the target object is the trajectory feature. When the trajectory feature value is zero, the target trajectory feature is met, and the grasping task is completed. Generally in Eq. (9), the target trajectory feature value can be interpreted as the desired value of the feature function, denoted as $\boldsymbol{C}_i^*$.

Based on the above definitions, our goal naturally becomes to learn a method that can make the trajectory feature value match the target trajectory feature value so that the corresponding trajectory of robotic arm can fulfill the task requirement. In Eq. (9), the general form of feature function has been given. Subsequently, some specific forms of the feature functions for several typical task scenarios with physical significance will be provided. It is noted that the feature functions for other tasks can also be designed in a similar manner, in which the form of $f_i(\boldsymbol{y}_t)$ should be designed according to the specific requirement of the task.

Feature function of obstacle avoidance task

A highly effective strategy for obstacle avoidance analysis is to create Boolean voxels of both the robot and the obstacles (Ratliff, Zucker, Bagnell, & Srinivasa, 2009), so that the signed Euclidean distance transform (EDT), the minimum distance between the robot's elements and the obstacles, can be calculated. In our case, the robotic links are enveloped with cylindrical bodies while the irregular obstacles with spherical bodies with a diameter equal to their maximum extensions, as shown in Fig. 2.

For the ith link l_i of an n DoFs robotic arm, the centroid of the link is aligned with the centroid of a cylindrical body. Then a function c_{l_i} related to obstacle avoidance can be expressed as:

$$c_{l_i} = \max\left\{\left(r_{l_i} + r_o + \varepsilon - d_{l_i}\right), 0\right\} \tag{10}$$

where r_{l_i} denotes the radius of the cylindrical body of link l_i, and r_o represents the radius of the spherical body of the obstacle. d_{l_i} is the Euclidean distance between the centroid of the cylindrical body and the centroid of the spherical body. The safety threshold is defined as ε.

Substituting Eq. (10) into Eq. (9), the trajectory feature corresponding to obstacle avoidance task is given by:

$$\begin{cases} \boldsymbol{C}_1 = \left[C_{l_1}, C_{l_2}, \ldots C_{l_n}\right]^{\mathrm{T}} \\ C_{l_i} = \int_{t=1}^{T} c_{l_i}\,\mathrm{d}t \end{cases} \tag{11}$$

From Eq. (11), it can be easily inferred that when C_{l_i} is greater than zero, it indicates that the ith link has collided with an obstacle. Conversely, when C_{l_i} equals zero, no collision has occurred. Hence, the target trajectory feature of collision avoidance task is defined as $\boldsymbol{C}_1^* = \left[C_{l_1}^*, C_{l_2}^*, \ldots, C_{l_n}^*\right]^{\mathrm{T}} = \boldsymbol{0}$.

Feature function of motion smoothness task

To ensure smooth movement, the reproduced trajectory of robotic arm is hoped to move along the demonstration trajectory as closely as possible. Then a Mahalanobis distance D_M between the trajectory $\boldsymbol{\Phi}_t\boldsymbol{\omega}$ and the demonstration trajectory distribution $\mathcal{N}\left(\boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Phi}_t\boldsymbol{\Sigma}_{\boldsymbol{\omega}}\boldsymbol{\Phi}_t^{\mathrm{T}}\right)$ is introduced. The Mahalanobis distance measures the similarity or difference between two high-dimensional vectors by taking into account the covariance relationship among their components, which can be calculated as:

$$\begin{aligned} &D_M\left(\boldsymbol{\Phi}_t\boldsymbol{\omega}, \boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}\right) \\ &= \sqrt{\left(\boldsymbol{\Phi}_t\boldsymbol{\omega} - \boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}\right)^{\mathrm{T}}\left(\boldsymbol{\Phi}_t\boldsymbol{\Sigma}_{\boldsymbol{\omega}}\boldsymbol{\Phi}_t^{\mathrm{T}}\right)^{-1}\left(\boldsymbol{\Phi}_t\boldsymbol{\omega} - \boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}\right)} \end{aligned} \tag{12}$$

Accordingly, the trajectory feature in motion smoothness task is designed as follows:

$$C_2 = \int_{t=1}^{T} D_M\left(\boldsymbol{\Phi}_t\boldsymbol{\omega}, \boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}\right)\mathrm{d}t \tag{13}$$

Regarding the target trajectory feature of the motion smoothness task, the minimum trajectory feature value for motion smoothness of the demonstration trajectories is extracted as the target trajectory feature value, which can be expressed as follows:

$$\begin{cases} C_2^* = \min\left(D_i - \min(D)/\max(D) - \min(D)\right) \\ D_i = \int_{t=1}^{T} D_M\left(\boldsymbol{\Phi}_t\boldsymbol{\omega}_i, \boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}\right)\mathrm{d}t \end{cases} \tag{14}$$

with $D = \left\{D_i\right\}_{i=1}^{M}$ the set of trajectory features for motion smoothness of the demonstration trajectories.

Feature function of joint state modulation task

In the scenario of joint state modulation task, the joints of the robotic arm are required to reach the desired state at a given time. In other words, the robotic arm needs to achieve the expected configuration at a given time. Therefore, the current state of the joint trajectory at the given time step is defined as the trajectory feature in joint state modulation task, that is:

$$\boldsymbol{C}_3 = \boldsymbol{\Phi}_{t^*}\boldsymbol{\omega} \tag{15}$$

with t^* the expected time step at which the joint modulation is executed.

Naturally, the target trajectory feature of the joint state modulation task is defined as the desired state of the joint at the given time step, represented as $\boldsymbol{C}_3^* = \boldsymbol{y}_t^*$.

3.2. Extrapolation learning of single task

In this section, we aim to extrapolate a satisfactory trajectory for task completion from existing demonstration trajectories through the conditional probability calculation to modulate the trajectory feature to the target trajectory feature.

Based on the feature model of a specific task, a corresponding set of trajectory feature data samples can be calculated from the demonstration trajectories, denoted as $\left\{\boldsymbol{C}^i\right\}_{i=1}^{M}$, where $\boldsymbol{C}^i \in \mathbb{R}^D$ is the trajectory feature of the ith demonstration trajectory and M is the number of demonstrations. Similar to the weight distribution in ProMP (see Eq. (3)), these trajectory features can also be modeled as a multi-dimensional Gaussian distribution, that is:

$$p\left(\boldsymbol{C}; \boldsymbol{\theta}_{\boldsymbol{C}}\right) = \mathcal{N}\left(\boldsymbol{C}\,|\boldsymbol{\mu}_{\boldsymbol{C}}, \boldsymbol{\Sigma}_{\boldsymbol{C}}\right) \tag{16}$$

where $\boldsymbol{\mu}_{\boldsymbol{C}} \in \mathbb{R}^D$ and $\boldsymbol{\Sigma}_{\boldsymbol{C}} \in \mathbb{R}^{D\times D}$ respectively denote the mean vector and covariance matrix of the trajectory feature distribution. The distribution parameters can be calculated using MLE as follows:

$$\begin{cases} \boldsymbol{\mu}_{\boldsymbol{C}} = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{C}^i \\ \boldsymbol{\Sigma}_{\boldsymbol{C}} = \frac{1}{M}\sum_{i=1}^{M}\left(\boldsymbol{C}^i - \boldsymbol{\mu}_{\boldsymbol{C}}\right)\left(\boldsymbol{C}^i - \boldsymbol{\mu}_{\boldsymbol{C}}\right)^{\mathrm{T}} \end{cases} \tag{17}$$

Furthermore, with the weight distribution and trajectory feature distribution corresponding to the demonstration trajectories, a probability correlation between weight and trajectory feature can be established through a joint Gaussian distribution model, which can be expressed as:

$$p(\boldsymbol{\omega}, \boldsymbol{C}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_{\boldsymbol{\omega}} \\ \boldsymbol{\mu}_{\boldsymbol{C}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}} & \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{C}} \\ \boldsymbol{\Sigma}_{\boldsymbol{C}\boldsymbol{\omega}} & \boldsymbol{\Sigma}_{\boldsymbol{C}\boldsymbol{C}}\end{bmatrix}\right) \tag{18}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{\Sigma}_{\boldsymbol{C}\boldsymbol{C}} \in \mathbb{R}^{D\times D}$ represent the covariance matrices of weights and trajectory features themselves, respectively. $\boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{C}} \in \mathbb{R}^{N\times D}$ and $\boldsymbol{\Sigma}_{\boldsymbol{C}\boldsymbol{\omega}} \in \mathbb{R}^{D\times N}$ are the covariance matrices between weights and trajectory features, which can also be obtained through MLE on the combination data $\left\{\left[\left(\boldsymbol{\omega}^i\right)^{\mathrm{T}}, \left(\boldsymbol{C}^i\right)^{\mathrm{T}}\right]^{\mathrm{T}}\right\}_{i=1}^{M}$.

Subsequently, applying Bayes' theorem allows us to match the trajectory feature to the target trajectory feature and infer weight from the joint Gaussian distribution. By calculating the conditional probability, the posterior distribution of weight given the target trajectory feature value condition can be obtained, and hence the posterior distribution of weight can be calculated as:

$$\begin{aligned}\pi\left(\boldsymbol{\omega}^*\right) &= p\left(\boldsymbol{\omega}\,|\boldsymbol{C}\right)|_{\boldsymbol{C}=\boldsymbol{C}^*} \\ &= \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\omega}}+\boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{C}}\boldsymbol{\Sigma}_{\boldsymbol{C}\boldsymbol{C}}^{-1}\left(\boldsymbol{C}^*-\boldsymbol{\mu}_{\boldsymbol{C}}\right), \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}}-\boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{C}}\boldsymbol{\Sigma}_{\boldsymbol{C}\boldsymbol{C}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{C}\boldsymbol{\omega}}\right) \\ &= \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\omega}}^*, \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}}^*\right)\end{aligned} \tag{19}$$

It is observed that the resulting weight distribution is still a conditional Gaussian distribution. Furthermore, according to the aforementioned reproduction formulas in ProMP (see Eqs. (4) and (5)), substituting the Eq. (19) into Eq. (4), the posterior distribution of joint state at time step t can be obtained as:

$$\begin{aligned}p\left(\boldsymbol{y}_t^*; \boldsymbol{\theta}_{\boldsymbol{\omega}}^*\right) &= \int \mathcal{N}\left(\boldsymbol{y}_t \left| \boldsymbol{\Phi}_t\boldsymbol{\omega}, \boldsymbol{\Sigma}_y\right.\right)\mathcal{N}\left(\boldsymbol{\omega}\,|\boldsymbol{\mu}_{\boldsymbol{\omega}}^*, \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}}^*\right)\mathrm{d}\boldsymbol{\omega} \\ &= \mathcal{N}\left(\boldsymbol{y}_t \left| \boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{\omega}}^*, \boldsymbol{\Phi}_t\boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}}^*\boldsymbol{\Phi}_t^{\mathrm{T}}+\boldsymbol{\Sigma}_y\right.\right)\end{aligned} \tag{20}$$

Subsequently, substituting Eq. (20) into Eq. (5), the posterior distribution of joint trajectory is obtained as:

$$p\left(\boldsymbol{\xi}^*; \boldsymbol{\theta}_{\boldsymbol{\omega}}^*\right) = \prod_T p\left(\boldsymbol{y}_t^*; \boldsymbol{\theta}_{\boldsymbol{\omega}}^*\right) \tag{21}$$

The final reproduced joint trajectory is generated through sampling from the posterior trajectory distribution. The trajectory distribution represents the probability of sampled trajectories with feature values matching the target trajectory feature value. In other words, it indicates the likelihood of the sampled trajectories being able to accomplish the specific task. Therefore, there may be multiple trajectories that can satisfy the task requirement. Typically, we select the joint trajectory with the highest sampling probability, i.e., the mean joint trajectory, as the final reproduced trajectory. The mechanism about the proposed conditional probability method to achieve extrapolation can be explained by Fig. 3.

In this research, the demonstration trajectories are modeled as the probability distribution. Thus, the range of the initial demonstration region can be represented by the distribution and any trajectory covered by this distribution can be easily regenerated. In Fig. 3, the demonstration data is described by a probability distribution (μ, σ) as shown in Fig. 3a, or more intuitively, by a trajectory distribution as depicted in Fig. 3b. However, due to the limited demonstration, trajectories with specific task requirements may be out of the distribution, for instance, the yellow star in Fig. 3a or the corresponding red line in Fig. 3b. To complete these tasks, a wider distribution or region to cover the trajectory is required. Then a modification to the demonstration data distribution is conducted as in Eq. (19). The trajectory feature is modulated to the target trajectory feature through the conditional probability calculation, and the weight distribution is adjusted. As a consequence, the initial distribution (μ, σ) is changed to (μ^*, σ^*). The trajectory that can satisfy the task, which is out of the initial distribution (μ, σ) in Fig. 3a and out of the demonstration trajectory distribution in Fig. 3b, is now in the range of the new distribution (μ^*, σ^*) in Fig. 3c and the expanded area corresponding to Fig. 3d. Thus, the trajectory that satisfies the task is reproducible and the task can be completed.

3.3. Rapid learning of multiple tasks

In this section, the single-task learning is extended to the multi-task learning. In ProMP, when multiple joint state points are required reaching, the conditioning modulation in Eq. (7) can be iteratively applied as follows (Maeda et al., 2017):

$$\begin{cases}\boldsymbol{\mu}_{\boldsymbol{\omega}}^{[i+1]} = \boldsymbol{\mu}_{\boldsymbol{\omega}}^{[i]} + \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i]}\boldsymbol{\Phi}_{t_i^*}^{\mathrm{T}}\left({}^i\boldsymbol{\Sigma}_y^* + \boldsymbol{\Phi}_{t_i^*}\boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i]}\boldsymbol{\Phi}_{t_i^*}^{\mathrm{T}}\right)^{-1}\left(\boldsymbol{y}_{t_i}^* - \boldsymbol{\Phi}_{t_i^*}\boldsymbol{\mu}_{\boldsymbol{\omega}}^{[i]}\right) \\ \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i+1]} = \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i]} - \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i]}\boldsymbol{\Phi}_{t_i^*}^{\mathrm{T}}\left({}^i\boldsymbol{\Sigma}_y^* + \boldsymbol{\Phi}_{t_i^*}\boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i]}\boldsymbol{\Phi}_{t_i^*}^{\mathrm{T}}\right)^{-1}\boldsymbol{\Phi}_{t_i^*}\boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i]}\end{cases} \tag{22}$$

where $\boldsymbol{\mu}_{\boldsymbol{\omega}}^{[i]}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\omega}}^{[i]}$ are the distribution parameters of the ith iterative step. $\boldsymbol{y}_{t_i}^*$ represents the ith desired joint state point at the given time step t_i^*.

From Eq. (22), it can be observed that this formula involves repeatedly updating the conditional distribution parameters until incorporating all joint state points. The reproduced joint trajectory reaches the desired state one by one at each given time step through iterative modulation. A significant advantage of this process is the modulation independence, which means the previous modulation will not be affected by the current modulation. This enables the generalization from modulating multiple joint state points to modulating multiple trajectory features when the trajectory feature of each single task is viewed as a joint state point on the trajectory.

Thus, similar to Eq. (22), the iterative formula for multiple trajectory features' modulation is derived by rewriting Eq. (19) as follows:

$$\begin{cases}\boldsymbol{\mu}_{\boldsymbol{\omega}}^{[i+1]} = \boldsymbol{\mu}_{\boldsymbol{\omega}}^{[i]} + \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{C}_i}\boldsymbol{\Sigma}_{\boldsymbol{C}_i\boldsymbol{C}_i}^{-1}\left(\boldsymbol{C}_i^* - \boldsymbol{\mu}_{\boldsymbol{C}_i}\right) \\ \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}}^{[i+1]} = \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}}^{[i]} - \boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{C}_i}\boldsymbol{\Sigma}_{\boldsymbol{C}_i\boldsymbol{C}_i}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{C}_i\boldsymbol{\omega}}\end{cases} \tag{23}$$

with $\boldsymbol{C}_i, \boldsymbol{C}_i^*$ representing the trajectory feature and target trajectory feature of the ith task. $\boldsymbol{\mu}_{\boldsymbol{\omega}}^{[i]}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\omega}\boldsymbol{\omega}}^{[i]}$ are the mean vector and covariance matrix of weight distribution in the ith iterative step, respectively. Through repeated modulation and sequential substitution of each target trajectory feature into Eq. (23), all task requirements can be fulfilled. The procedure can be illustrated in Figs. 4 and 5.

As shown in Fig. 4, assuming that Q tasks are required to be completed, the trajectory feature for these tasks are represented as $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_Q$, with their respective target trajectory feature as $\boldsymbol{C}_1^*, \boldsymbol{C}_2^*, \ldots, \boldsymbol{C}_Q^*$. The first modulation calculates the conditional probability $p\left(\boldsymbol{\omega}\right)\left|_{\boldsymbol{C}_1=\boldsymbol{C}_1^*}\right.$, i.e., the weight probability $p\left(\boldsymbol{\omega}\right)$ given the condition that the target trajectory feature of the first task $\boldsymbol{C}_1^*$ is required, and obtain the posterior distribution $p\left(\boldsymbol{\omega}\left|\boldsymbol{C}_1^*\right.\right)$. Then, the satisfactory trajectory can be regenerated by the modulated distribution, and the first task can be completed. Subsequently, moving to the second task, the second modulation calculates the conditional probability $p\left(\boldsymbol{\omega}\left|\boldsymbol{C}_1^*\right.\right)\left|_{\boldsymbol{C}_2=\boldsymbol{C}_2^*}\right.$, i.e., the weight probability $p\left(\boldsymbol{\omega}\left|\boldsymbol{C}_1^*\right.\right)$ under the condition of the target trajectory feature for the second task $\boldsymbol{C}_2^*$. It adds an additional conditional probability calculation on top of the original conditional probability $p\left(\boldsymbol{\omega}\left|\boldsymbol{C}_1^*\right.\right)$. Not only the second target trajectory feature is satisfied, but also the tuned results to complete the first task is not ruined. In other words, the second modulation is conditionally calculated with the pre-requirement of the first task can be met, i.e., a conditional probability $p\left(\boldsymbol{\omega}\left|\boldsymbol{C}_1^*, \boldsymbol{C}_2^*\right.\right)$. Repeating this procedure through Q modulations, the posterior weight distribution is obtained as $p\left(\boldsymbol{\omega}\left|\boldsymbol{C}_1^*, \boldsymbol{C}_2^*, \ldots, \boldsymbol{C}_Q^*\right.\right)$. Then the final posterior distribution covering all tasks is derived as $p\left(\boldsymbol{\xi}^*\right)\left|_{\boldsymbol{C}_1=\boldsymbol{C}_1^*, \boldsymbol{C}_2=\boldsymbol{C}_2^*, \ldots, \boldsymbol{C}_Q=\boldsymbol{C}_Q^*}\right.$ using Eqs. (20) and (21). This procedure is also illustrated in Fig. 5.

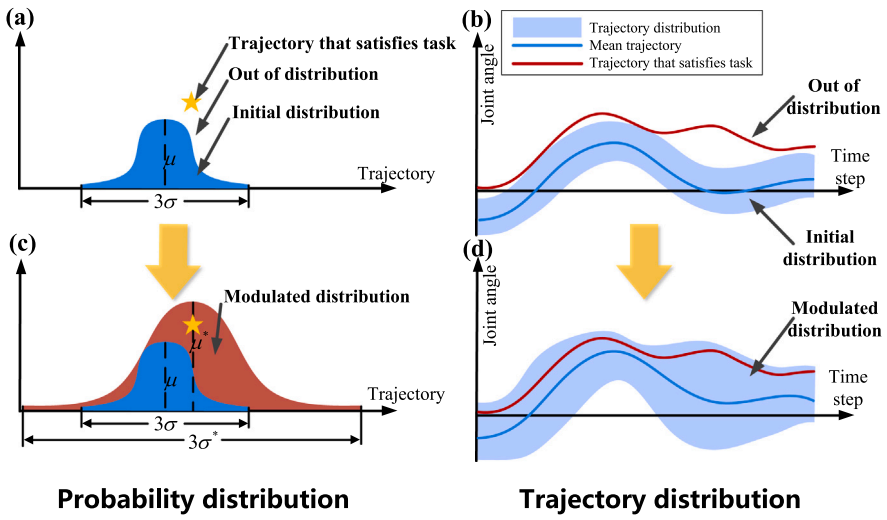

Fig. 3. Diagram of extrapolation in the single task, which respectively illustrates the transitions in probability distribution of trajectory and the corresponding trajectory distribution through conditional probability computation.

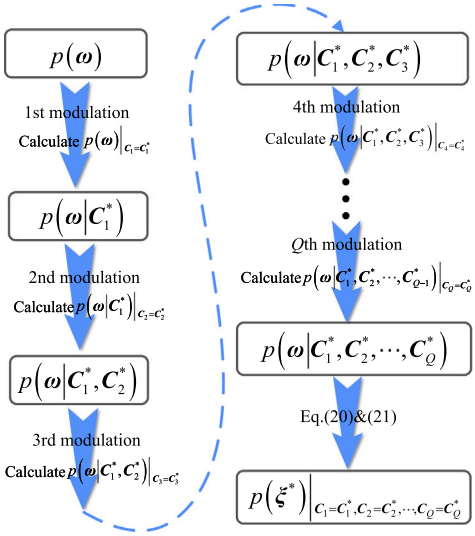

Fig. 4. Illustrative diagram of probabilistic theory underlying multi-task modulation.

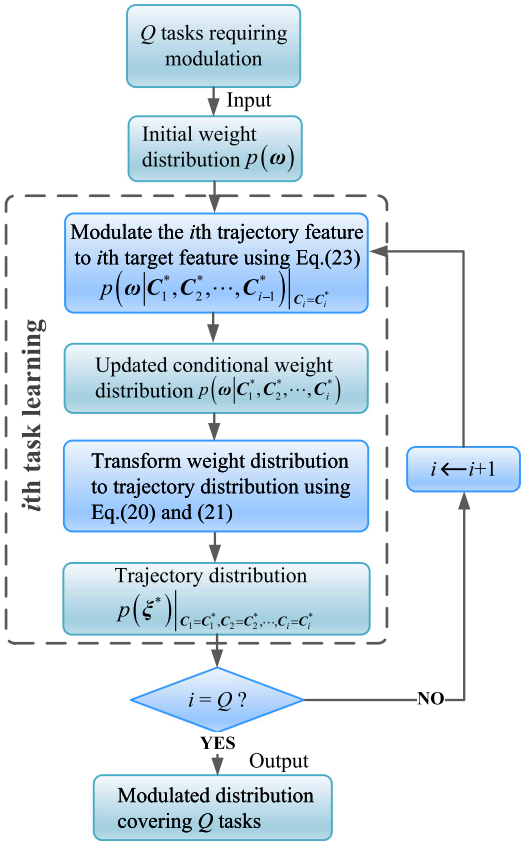

Fig. 5. Multi-task learning process.

4. Experiment and analysis

In this section, simulation and experiment will be conducted to verify the effectiveness of the proposed framework.

4.1. Simulation and experiment

The feature models of obstacle avoidance, motion smoothness and joint state modulation tasks are adopted. The experiment environment is designed as a scenario where a 7-DoF Franka-Emika robot is required to learn lifting and handling skills, grasping an object from an initial position and transporting it to a specified destination while avoiding the obstacle and maintaining smooth movement. The robot is firstly demonstrated through kinesthetic teaching to acquire trajectories of joint angles and angular velocities. The demonstration data, consisting of $M = 30$ instances with $T = 150$ time steps, is then encoded using a total of $N = 60$ Gaussian basis functions. Here, the two configurations of robotic arm for grasping and placing object are chosen as the desired joint state points for the joint state modulation task at the 60th and 150th steps, respectively. It is worth noting that the obstacle is absent during the demonstration stage while the robot is required to avoid it in the test.

Validation of multi-task learning

Employing the proposed framework, the reproduced behavior of the Franka-Emika robot is presented in Fig. 6 and the transition of each joint trajectory is depicted in Fig. 7.

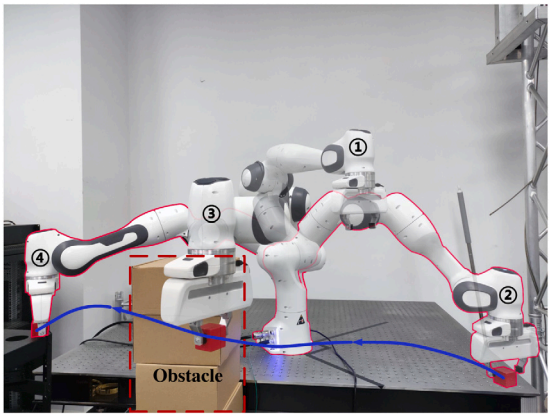

Fig. 6. Reproduced trajectory of Franka-Emika robot. The transparency illustrates the temporal transition of the robot's behavior from light to dark. Configuration ① represents the initial configuration of the robotic arm, Configuration ② corresponds to the target configuration of the 1st desired joint state point where the arm reaches the object grasping position. Configuration ③ shows the robot's obstacle avoidance behavior, and Configuration ④ represents the target configuration at the 2nd joint state point where the arm reaches the object placing position.

Table 1
Error between desired and actual values of two target configurations at the given time steps.

Joint	The grasping target configuration (rad)		The placing target configuration (rad)		Relative error
	Actual	Desired	Actual	Desired	
1	0.3147	0.3127	−1.9232	−1.8867	0.21% 3.66%
2	0.8568	0.8577	1.0501	1.0480	0.08% 0.21%
3	0.1752	0.1738	0.1553	0.2001	0.14% 4.48%
4	−1.6027	−1.6076	−0.8636	−0.8609	0.49% 0.27%
5	−0.2071	−0.2181	−0.1331	−0.1427	1.09% 1.17%
6	2.4264	2.4242	1.8983	1.9037	0.22% 0.54%
7	1.7076	1.6997	−2.4500	−2.4994	0.79% 4.94%

It is evident from Fig. 6 that the robotic arm successfully avoids the obstacle while completing the transportation task. Additionally, Fig. 7 indicates that the robotic arm can accurately reach two desired joint state points, which correspond to the grasping and placing target configurations in Fig. 6. The error between the desired and actual values of two target configurations at given time steps is presented in Table 1. The relative error can be kept at a low level not exceeding 5%, typically satisfying the joint state modulation task. Moreover, despite the noticeable fluctuation in joint trajectory around the 100th time step due to the obstacle avoidance behavior, the posterior joint trajectory closely follows the mean trajectory. These findings suggest that the reproduced trajectory of the robotic arm meets all the requirements, i.e., obstacle avoidance, motion smoothness and joint state modulation.

Validation of extrapolation learning

To evaluate the extrapolation performance of the proposed method, it is observed in Fig. 7 that the 2nd desired joint state points of the 2nd, 4th and 6th joints of the robotic arm fall outside the demonstration area (as indicated by the state points enclosed by the red circle markers). This indicates that the required configuration has exceeded the demonstration area, but the reproduced trajectory is still modulated to reach these points. This demonstrates that the learned action has already surpassed teaching and achieved extrapolation.

Meanwhile, the trajectory features for obstacle avoidance task of both the demonstration trajectories and the posterior reproduced trajectory are also extracted, as shown in Fig. 8. In the feature model designed for the obstacle avoidance task, the feature value of trajectory is greater than zero when the robotic arm collides with the obstacle, otherwise, it remains zero. Notably, in Fig. 8, even when all demonstration trajectories result in collisions with the obstacle (as indicated by a non-zero value for all gray curves around the 100th time step), the reproduced trajectory is still modulated to avoid the obstacle (as represented by the red curve remaining at zero value throughout). This also demonstrates that the robotic arm has acquired obstacle avoidance skill beyond the teaching region.

Case of multiple task requirements conflict

Let us consider a situation where the desired configuration is within the obstacle area, i.e., there is a conflict between the joint state modulation task and the obstacle avoidance task. The end-effector position of the 1st desired configuration (object grasping location) is set as $[0.61, 0.30, 0.01]^{\mathrm{T}}$, and the centroid position of the 1st obstacle is set as $[0.60, 0.15, 0.10]^{\mathrm{T}}$. The end-effector position of the 2nd desired configuration (object placing location) is set as $[-0.16, -0.78, 0.15]^{\mathrm{T}}$, and the centroid position of the 2nd obstacle is set as $[-0.1, -0.58, 0.1]^{\mathrm{T}}$. The envelope sphere radius of both obstacles are 0.1 m. The resulting posterior behavior of the robotic arm is illustrated in Fig. 9.

From Fig. 9, the robotic arm successfully completes the joint state modulation task, reaching the desired configuration. However, it fails to fully accomplish the obstacle avoidance task, as it avoids the blue obstacle but collides with the red one. It is found that the result is related to the order of multi-task modulation. When multiple tasks conflict with each other, the proposed approach prioritizes satisfying the requirement of the task in the most recent modulation, while disregarding the tasks in the previous modulation. Therefore, it is suggested that the important task should be modulated later than other tasks. In this case, three tasks are modulated in the following order: motion smoothness task at the lowest priority (first task), obstacle avoidance task at the intermediate priority (second task) and joint modulation task at the highest priority (third task). Therefore, the final reproduced trajectory fulfills the highest priority task but falls short of completing the intermediate priority task. The motion smoothness, as the lowest priority task, is lastly satisfied. As seen in Fig. 7, when encountering an obstacle around the 100th time step, the reproduced trajectory first satisfies the obstacle avoidance requirement, rather than continuing to maintain motion smoothness by following the mean demonstration trajectory.

4.2. Comparison

The ProMP based Trajectory Planning (PROMPT) method integrates ProMP and Stochastic Optimization Algorithm (STOMP), utilizing the learned distribution from ProMP to sample a series of noisy trajectories to explore the surrounding space, eventually obtain an updated trajectory by iteratively optimizing a task-related cost function (Kalakrishnan, Chitta, Theodorou, Pastor, & Schaal, 2011; Löw, Bandyopadhyay, Williams, & Borges, 2021). In this section, we proceed to compare with PROMPT to demonstrate the performance of the proposed framework.

Using the lifting and handling scenario described above, we equip both methods with identical demonstration data and task functions. Specifically, PROMPT is set to generate 20 noisy trajectories in each iteration with a maximum of 100 iterations. If the maximum iteration count is exceeded, it is considered a failure. Also, if the final trajectory of robotic arm fails to avoid obstacles, it will also be deemed as a failure. A total of 200 repeated tests are conducted by varying the parameters of the task scenario, including the grasping and placing positions of the transportation, obstacle positions and number of obstacles. Here the robot's behaviors in typical scenarios with two and three obstacles using the proposed approach are presented in Fig. 10. The comparative results in terms of running time, joint state points' error and success rate are shown in Fig. 11.

Fig. 11(a) illustrates the superiority of the proposed method over iteration based learning approach in terms of computational efficiency.

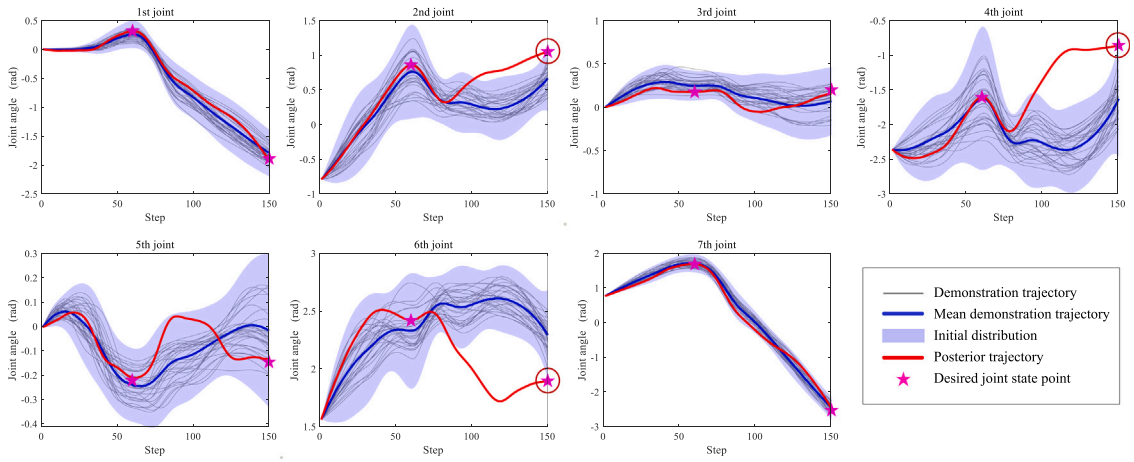

Fig. 7. Transition of joint trajectory of Franka-Emika robot. The blue curve denotes the mean trajectory of demonstration trajectory distribution, and the blue shaded area represents the demonstration trajectory distribution. The red curve represents the reproduced trajectory obtained from multi-task learning, and the star marker indicates the desired joint state point traversed by the reproduced trajectory. The red circle marker indicates that the joint state point has exceeded the demonstration region.

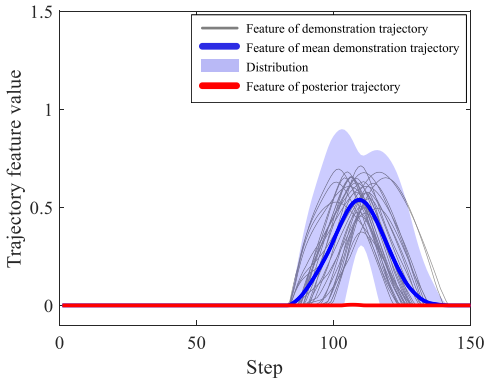

Fig. 8. Trajectory features for obstacle avoidance task of Franka-Emika robot.

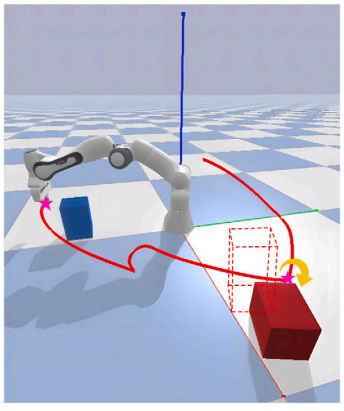

Fig. 9. Reproduced trajectory of Franka-Emika robot when the joint state modulation task conflicts with the obstacle avoidance task. The end-effector positions of the desired configurations are marked by the red star markers.

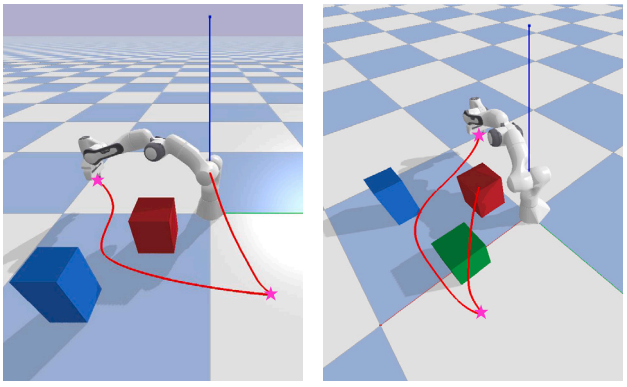

Fig. 10. Simulation in scenarios with multiple obstacles. (a) Two obstacles. (b) Three obstacles.

This disparity can be attributed to PROMPT's need for repeated computation of the task function during each iteration, which is computationally inefficient, particularly for complex cost models. In contrast, the proposed method adopts a computation-replace-iteration methodology and calculates the task function once to enhance the learning efficiency. Figs. 11(b) and 11(c) display the error at the desired joint state points of two methods. It can be observed that the proposed method and PROMPT exhibit errors of similar magnitude. However, PROMPT has more outliers (as indicated by the red plus markers), particularly at the first joint state point, where the maximum outlier is several tens of times larger than the average level. This is due to its inherent exploration mechanism, which introduces noisy trajectories with a certain degree of randomness, leading to undesirable results. In contrast, the proposed method shows greater stability and does not exhibit as many outliers. In terms of the success rate as seen in Fig. 11(d), both methods have matched success rates. Analysis of failed cases reveals that the main reason of PROMPT's failure is the randomness in policy search, which sometimes prevents the discovery of better policies and exceeds the maximum number of iterations. The main cause of failure in the proposed method lies in occasional conflicts between multiple tasks due to improper task parameter settings, leading to bad results such as unsuccessful obstacle avoidance.

Although the proposed method performs remarkable advantages in terms of learning efficiency and performance, it has a major drawback. The solutions obtained from LfD-based methods tend to be feasible rather than optimal compared to the iteration-based learning methods. This should be further investigated.

5. Conclusion and future work

In this paper, a novel skill learning framework for industrial robot in multi-task scenario is proposed, which leverages a task-parameterized learning approach based on ProMP to achieve multi-task learning,

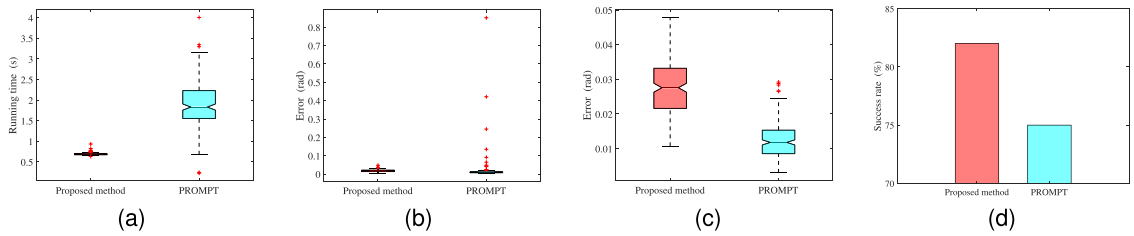

Fig. 11. Comparison results with PROMPT. (a) Running time. (b) Error of the 1st joint state point. (c) Error of the 2nd joint state point. (d) Success rate.

a computation-replace-iteration methodology to enhance the learning efficiency, and a conditional probability calculation to enable the robot to learn beyond teaching. These characteristics make the framework particularly suitable for complex industrial production and manufacturing tasks under the condition of limited training data. Additionally, the effectiveness and performance of the proposed framework have been confirmed through both simulation and real robot experiment. It has the potential to inspire the development of skills learning for future industrial robot operations.

In the future work, we consider overcoming the limitation of the feasible solutions by combining the proposed method with the lightweight Reinforcement Learning framework. It is expected to address the issue of high computational efficiency but only obtaining feasible solutions in LfD, as well as the time-consuming problem of seeking optimal solutions in RL. Specifically, the learned distribution obtained through the proposed method is used as the initial policy, and the lightweight RL method is employed for further policy improvement with the fewer computational resources. With the prior knowledge, significant performance improvements could be achieved with a small number of learning steps.

CRediT authorship contribution statement

Chengfei Yue: Project administration, Funding acquisition, Conceptualization. **Tian Gao:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology. **Lang Lu:** Writing – review & editing, Visualization. **Tao Lin:** Software, Investigation. **Yunhua Wu:** Conceptualization.

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.

Data availability

Data will be made available on request.

Acknowledgments

This work has been supported by National Natural Science Foundation of China (12372045), Shenzhen Science and Technology Program, China (JCYJ20220818102207015), and Guangdong Basic and Applied Basic Research Foundation, China (2023B1515120018).

References

Akbulut, M., Oztop, E., Seker, M. Y., Hh, X., Tekden, A., & Ugur, E. (2021). Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing. In *Conference on robot learning* (pp. 1896–1907). PMLR.

Allen, T. T., Roychowdhury, S., & Liu, E. (2018). Reward-based Monte Carlo-Bayesian reinforcement learning for cyber preventive maintenance. *Computers & Industrial Engineering*, *126*, 578–594.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

Burlizzi, R., Vochten, M., De Schutter, J., & Aertbeliën, E. (2022). Extending extrapolation capabilities of probabilistic motion models learned from human demonstrations using shape-preserving virtual demonstrations. In *2022 IEEE/RSJ international conference on intelligent robots and systems* (pp. 10772–10779). IEEE.

Calinon, S., Evrard, P., Gribovskaya, E., Billard, A., & Kheddar, A. (2009). Learning collaborative manipulation tasks by demonstration using a haptic interface. In *2009 international conference on advanced robotics* (pp. 1–6). IEEE.

Chatzis, S. P., Korkinof, D., & Demiris, Y. (2012). A nonparametric Bayesian approach toward robot learning by demonstration. *Robotics and Autonomous Systems*, *60*(6), 789–802.

Elliott, S., Xu, Z., & Cakmak, M. (2017). Learning generalizable surface cleaning actions from demonstration. In *2017 26th IEEE international symposium on robot and human interactive communication (RO-MAN)* (pp. 993–999). IEEE.

Flash, T., & Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, *15*(6), 660–666.

Gomez-Gonzalez, S., Neumann, G., Schölkopf, B., & Peters, J. (2020). Adaptation and robust learning of probabilistic movement primitives. *IEEE Transactions on Robotics*, *36*(2), 366–379.

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.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation* (pp. 4569–4574). IEEE.

Khansari-Zadeh, S. M., & Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, *27*(5), 943–957.

Kober, J., Gienger, M., & Steil, J. J. (2015). Learning movement primitives for force interaction tasks. In *2015 IEEE international conference on robotics and automation* (pp. 3192–3199). IEEE.

Kormushev, P., Calinon, S., & Caldwell, D. G. (2011). Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, *25*(5), 581–603.

Lin, Y., Ren, S., Clevenger, M., & Sun, Y. (2012). Learning grasping force from demonstration. In *2012 IEEE international conference on robotics and automation* (pp. 1526–1531). IEEE.

Löw, T., Bandyopadhyay, T., Williams, J., & Borges, P. V. (2021). PROMPT: Probabilistic motion primitives based trajectory planning. In *Robotics: science and systems*.

Maeda, G. J., Neumann, G., Ewerton, M., Lioutikov, R., Kroemer, O., & Peters, J. (2017). Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks. *Autonomous Robots*, *41*, 593–612.

Mussa-Ivaldi, F. A. (1999). Modular features of motor control and learning. *Current Opinion in Neurobiology*, *9*(6), 713–717.

O'Hogan, A., & Foster, J. (1999). Kendall's advanced theory of statistics, Bayesian inference. *Arnold, London*.

Paraschos, A., Daniel, C., Peters, J. R., & Neumann, G. (2013). Probabilistic movement primitives. *Advances in Neural Information Processing Systems*, *26*.

Paraschos, A., Daniel, C., Peters, J., & Neumann, G. (2018). Using probabilistic movement primitives in robotics. *Autonomous Robots*, *42*, 529–551.

Park, E. H., Ntuen, C. A., Vootla, S., & Park, Y. (1994). Adaptive learning of human motion by a telerobot using a neural network model as a teacher. *Computers & Industrial Engineering*, *27*(1–4), 453–456.

Peng, S., et al. (2021). Reinforcement learning with Gaussian processes for condition-based maintenance. *Computers & Industrial Engineering*, *158*, Article 107321.

Perez-Villeda, H., Piater, J., & Saveriano, M. (2023). Learning and extrapolation of robotic skills using task-parameterized equation learner networks. *Robotics and Autonomous Systems*, *160*, Article 104309.

Rana, M., Mukadam, M., Ahmadzadeh, S. R., Chernova, S., & Boots, B. (2018). Towards robust skill generalization: Unifying learning from demonstration and motion planning. In *Intelligent robots and systems*.

Ratliff, N., Zucker, M., Bagnell, J. A., & Srinivasa, S. (2009). CHOMP: Gradient optimization techniques for efficient motion planning. In *2009 IEEE international conference on robotics and automation* (pp. 489–494). IEEE.

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., Calinon, S., Caldwell, D., Jiménez, P., & Torras, C. (2013). Learning collaborative impedance-based robot behaviors. *Vol. 27*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 1422–1428). (1).

Rozo, L., Calinon, S., Caldwell, D. G., Jimenez, P., & Torras, C. (2016). Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, *32*(3), 513–527.

Saveriano, M., Abu-Dakka, F. J., Kramberger, A., & Peternel, L. (2021). Dynamic movement primitives in robotics: A tutorial survey. arXiv preprint arXiv:2102.03861.

Schaal, S. (1996). Learning from demonstration. *Advances in Neural Information Processing Systems*, *9*.

Schaal, S. (2006). Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. *Adaptive Motion of Animals and Machines*, 261–280.

Ude, A., Gams, A., Asfour, T., & Morimoto, J. (2010). Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, *26*(5), 800–815.

Ying, K.-C., Pourhejazy, P., Cheng, C.-Y., & Cai, Z.-Y. (2021). Deep learning-based optimization for motion planning of dual-arm assembly robots. *Computers & Industrial Engineering*, *160*, Article 107603.

Zhou, Y., Gao, J., & Asfour, T. Learning Via-Point Movement Primitives with Inter- and Extrapolation Capabilities. in 2019 IEEE, in RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4301–4308.