A Dynamic Movement Primitives-Based Tool Use Skill Learning and Transfer Framework for Robot Manipulation

LU, Zhenyu <http://orcid.org/0000-0002-5446-7285>, WANG, Ning <http://orcid.org/0000-0002-3264-1852> and YANG, Chenguang <http://orcid.org/0000-0001-5255-5559>

A Dynamic Movement Primitives-based Tool Use Skill Learning and Transfer Framework for Robot Manipulation

Zhenyu Lu, *Member, IEEE*, Ning Wang, *Member, IEEE,* and Chenguang Yang*, Fellow, IEEE*

***Abstract*—This paper presents a framework for learning and transferring robot tool-use skills based on Dynamic Movement Primitives (DMPs) for robot fine manipulation. DMPs and their enhanced methods are employed to acquire a specific tool-use skill applicable to tools with similar sizes, shapes, and uses. However, the acquired skills may not be transferable to other scenarios and tools with variations. The new framework introduces two new types of skills based on DMPs: Object Operating (O2) skill and Tool Flipping (TF) skill. The O2 skill enables robots to handle tools for manipulating objects to achieve desired effects. The learning process for the O2 skill considers limitations imposed by tools and the environment during human demonstrations. Distinguishing between whether constraints can be modelled or not, we propose both a model-based and a constraint-based method to separate a constraint-irrelevant (CI) skill and the constrained conditions. The CI skill is generalized using a novel method called constrained -DMP lite, enabling adaptation to new tasks with special tools. The TF skill addresses situations where tools must generate an action to alter contacting positions on both objects and tools while avoiding conflicts during movement. Finally, the TF and O2 skills are generalized to be applied in creating a continuous action chain. We conduct several experiments to compare and analyze the advantages and disadvantages of the proposed methods with other approaches in terms of generalizability and calculation complexity.**

***Note to Practitioners*—Strengthening robot tool-use ability has been a hot research topic in recent years because these tools can extend the reachability and enhance the flexibility of robots. The previous research on DMPs has been utilized for learning tool-use skills. However, the learned skills few considered the tools' special use regulations, therefore the skill of using a tool is hard to transfer to another tool-use case. This paper explores tool-use skill learning and transfer between different tools by developing a framework based on the DMPs for this problem. The framework consists of two kinds of skills: O2 skill and TF skill with different purposes as well as a series of newly developed algorithms, such as constrained -DMP lite, a model-based and a constraint-based CI skill learning methods. These methods can separate the constraints from human demonstrations of using tools to achieve a CI skill and generalize the CI skill according to the constraints generated from a new tool-use manipulation task. We verify the effectiveness of the proposed framework through some typical tool-use experiments, including pushing objects, cutting and obstacle avoidance in actuality. The development of this framework can be used in industrial and house working scenarios.**

***Index Terms*—Dynamic movement primitives, Robot learning, Skill transfer, Multi-tool use skill, Robot manipulation**

I. Introduction

TOOLS are essential for extending the reach and enhancing the working efficiency of humans, which are also crucial for improving the flexibility of robot manipulations. As indicated in [1], tool use is an important avenue for the research of robot manipulation, and some preliminary studies have been conducted in this direction. Qin *et al.* made a comprehensive review of robot tool use and a classification as non-causal tool use and causal tool use. The causal tool use is further separated to single and multiple manipulations with detailed taxonomies [2]. Generally, a robot tool use skill is defined as manipulating an object through some actions to reach the desired objectives. Compared to robot causal tool use, non-causal tool use tasks emphasize generating accurate actions with less consideration of the objects, effects, and relations, which is applicable for the fine manipulations that require high precision, such as fastening nuts [4] and screws [5], pouring water [15], cutting and drilling objects [6], [7], [16], and drawing. Causal tool-use emphasizes actions of realizing the desired effects and the research has more widespread topics, covering tool recognition, selection, action planning, tool-use strategy and multiple tool-use cooperation in which several robots complete complicated and sequential tasks, such as tidying [17], assembling furniture [3], scooping using spoon-like tools [63] and manipulating deformable objects [65]. The majority of tool-use methodologies are verified by typical applications, including pushing-slider, pouring, cutting and peg -in-hole tasks [66].

The methodology of robot tool use can be categorized as tool-use recognition and tool-use skill generation. Tool recognition is to identify the "properties of a tool" [53] or "understanding tool-use desired effects" [52]. Understanding the affordances of objects was well-studied, but the functional usage recognition draws some attention but not been thoroughly researched [40]-[45] For example, Zhu *et al.* proposed a new framework aiming at understanding underlying functions, physics and causality in using objects as "tools" [41]. Qin *et al.* proposed KETO which can learn key point representations of tool-based manipulations

This work was supported in part by Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/S001913 and in part by the Marie Skłodowska-Curie Actions Individual Fellowship under Grant 101030691.

Zhenyu Lu and Ning Wang are with the Bristol Robotics Laboratory, University of the West of England, BS16 1QY, UK.

Chenguang Yang was with Bristol Robotics Laboratory, University of the West of England, Bristol, BS16 1QY, UK, and is with Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK.

*Corresponding author. Email:cyang@ieee.org

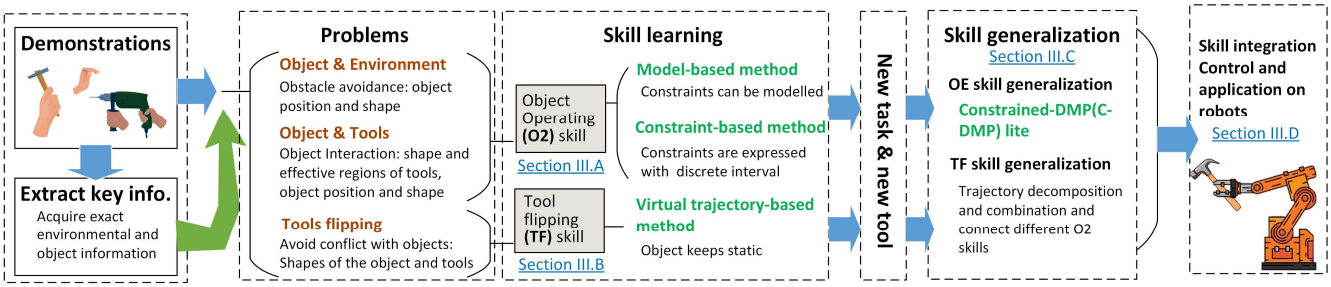


Fig. 1. Illustration of Tool-Use Skill Learning and Transfer Framework. The framework considers the relationships between objects and the environment, objects and tools, and tool-flipping operational requirements. The contributions are presented in skill learning and generalization modules and highlighted in green. The O2 skill is for action generation in manipulating objects and is divided into a model-based method and a constraint-based method. The TF skill is for generating tool flipping actions to change the tool contact regions. The two types of skills are integrated for robot tool-use manipulations.

[51]. Hassanin *et al.* classified three types of tool functionalities as motion-based, interaction-based, and shape-based functionalities, and the tool features are handcrafted and then trained by supervised learning [40]. However, there are limitations to the tool use, such as when objects or tools are significantly different from the training cases, it is difficult to process [45].

Robot tool-use skills can be generated using various methods, such as deep learning [62], [64], reinforcement learning [67], [68] and imitation learning [13], [14], [63] for robot arms [10], [11] and whole-body robots [12]. Imitation learning or learning from demonstration (LfD) technology drew much attention that enables robots to quickly acquire skills from demonstrations to mimic human tool-use actions safely by kinesthetic teaching, teleoperation, and passive observation [8], [9]. Usually, a long-term skill can be further segmented into several primitive skills for tool use that are easy to generalise and composed to achieve a new action [29]. Dynamic Movement Primitives (DMP) is a classic method of learning primitive skills that leverages the idea of attractors from dynamical systems [16], [19] to generate some actions for grasping [19]-[20], lower-limb prosthesis [21] and exoskeleton control [22]-[24], minimally invasive surgery [25], visual servoing control [26], and EMG-based impedance control [27]-[29]. The DMPs and their improved methods are categorized as non-causal tool use in [2]. However, it can be used to generate trajectories combined with deep learning [64] and reinforcement learning [68] for causal tool-use cases.

This paper will focus on DMP-based transferable causal tool-use skills, which are learned and transferred to another tool [2]. Feature matching of unlearned objects and known objects is a research branch in this direction as well as the prework for the following action generation algorithms. Tee and Brown *et. al,* did research on effective tool-use region recognition, which is the precursor to robot tool manipulation [52], [53], [56]. The research integrated various analytical models, deep learning and feature mapping methods with the prior knowledge of tools, objects, and environments to identify effective region features [13], [52] and key points [51] of tools and generate parametric constraints from demonstration [46]-[48]. In this paper, we will focus on tool-use action planning based on the tool recognition and perception results.

During the demonstration process, human actions or robot actions in kinesthetic teaching for a specific tool are recorded. However, the actions are specific to this tool-use case and only applicable to tools with similar sizes, shapes and usages, which may not be suitable for other tools. Meanwhile, humans easily transfer one tool-use skill to another tool, find the common use between the two tools and adapt actions to realize the same operational goals. In this paper, we refer to this human ability and name a skill, which is not constrained by functions, sizes and shapes of tools, as a constraint-irrelevant (CI) skill. For example, we use a hammer to strike a nail. The structure of the hammer and its special functions of holding the handle and using the hammer head to strike nails will constrain the hand's trajectory. For another tool, we can use it to realize the same purpose differently. Therefore, the CI skill is described as the 'strike' skill which is not limited to a hammer.

This paper aims to acquire CI skills from demonstrations by separating multiple constrained conditions from the tools, the environment and manipulation requirements and transferring the learned skills to another tool-use case. We propose the tool-use skill learning and transfer framework along with two types of skills based on the concept of CI skills: object operating (O2) skill and tool flipping (TF) skill in Fig. 1. O2 skill describes tool use trajectory affecting an object and enables the object to reach the design position. TF skill describes the process in which the object remains static and the tool changes contact state with the object or the ability to connect dispersed O2 skills to organize a continuous action. Meanwhile, due to the different constraints of the tools and the environment, we proposed a model-based and a constraint-based method for the O2 skills and a virtual-trajectory-based method for the TF skill.

The two O2 skills are distinguished by whether constraints can be modelled or expressed by discrete datasets. The model-based method contains a new iterative identification algorithm for estimating parameters of the model and the constraint-based method builds a Sigmoid-like function to extract CI skills from the trajectories with constraints. The TF skill divides a tool-use flipping actions into two correlative DMPs, one is the CI skill and the other is a virtual trajectory relating to the outline of the object. The learned CI skills can be generalized for a new task using a new tool under some constraints from the environment and the tool. The O2 skills are generalized by constrained-DMP lite, which is developed from a method in our previous research [21]. The TF skill generalization is realized by combining the generalized two correlative skills to achieve a new TF action to bridge the separated O2 skills to achieve a continuous tool-use action for new tools' applications. These skills are realized in different parts in Section III and marked in Fig.1 separately.

The remainder of this paper is organized as follows: Section 2 introduces the preliminary work, especially about DMPs and their improved methods. We present the limitations of using these methods for tool-use manipulation through an example. Section 3 presents the learning and generalization methods of the O2 skill and the TF skill and robot controllers to realize these skills in actuality. The generated trajectory is proved to converge to the target position, satisfies the rigorous constraints and requirements for tool-use operations and avoids conflicts with the environment and the objects. Section 4 conducts two experiments to verify the effectiveness of the proposed method integrating two types of skills. Section 5 makes a discussion on autonomy of skill learning and generalization, and increment of calculation complexity. Section 6 make a conclusion.

II. PRELIMINARY WORK

A. General DMP

Following the taxonomy of tool use in [2], DMP, proposed by Ijspeert *et al.* and updated in 2013 [18], is a typical non-causal tool use learning method. It has a concise and effective representation used for learning from demonstrations of robotic skills due to its strong generalization ability. A general DMP model is expressed as

$$\begin{cases} \tau \dot{v} = \alpha_z \left(\beta_z \left(g - x \right) - v \right) + f\left(s \right) \\ \tau \dot{x} = v \end{cases}, \tag{1}$$

where x is the position and $\dot{x}$ is the velocity of the trajectory, $f\left(s\right) = \mathbf{W}^T \mathbf{\Psi}(s)$ is a linear combination of n nonlinear Radial Basic Functions, named forcing function, which enables robots to follow any trajectory from the start x_0 to the goal g, where $\mathbf{W} = \left[w_1, w_2, ..., w_n \right]^T$, $\mathbf{\Psi}(s) = \left[\psi_1, \psi_2, ..., \psi_n \right]^T$ and

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

where c_i and $h_i > 0$ are the centers and widths of the Radial Basic Functions respectively, and $\alpha_z, \beta_z > 0$ are coefficients. Eq. (1) has the unique attracting point at $x = g, v = 0$. $\tau > 0$ is a scaling parameter, and s is a phase variable to achieve the dependency of $f\left(s\right)$ out of time, whose dynamics is expressed by a canonical system as

$$\tau \dot{s} = -\gamma s, \qquad \gamma > 0. \tag{3}$$

The converging time is modified by factor γ, and vector $\mathbf{W}$ is learned using locally weighted regression [29]:

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

where *N* represents the number of demonstrated trajectories and $f_j\left(s\right)$ represents the item calculated by the *jth* trajectory x_j, and f_j^{Tar} is the target value of $f\left(s\right)$:

$$f_j^{Tar} = \tau \dot{v}_j - \alpha_z \left(\beta_z \left(g - x_j \right) - v_j \right), \tag{5}$$

By changing the start x_0, goal g and scaling factor τ, the skill expressed by DMP can be generalized in a new task.

B. Improved DMP for constrained conditions

The DMPs model in (1) are improved to perform constrained tasks, such as obstacle avoidance [30] and cooperative manipulation [31], where constraints are generated by obstacles, and the relative distances of robot ends. The function of the DMP model is also modified. For example, Park *et. al* [32] and Tan *et. al* [38] added a gradient of a potential field to the forcing function. Hoffmann *et.al* [33] were motivated by the biological data and extended the DMP function into a new expression without singularities and large accelerations to achieve automatic goal adaptation and real-time obstacle avoidance. Umlauft *et. al* [35] and Gams *et. al* [37] developed coupling movement primitives (CMP) for cooperative manipulation. These improved DMPs are presented by adding a new term Δu to the forcing function $f\left(s\right)$, which can be expressed in a general form as

$$\begin{cases} \tau \dot{v} = \alpha_z \left(\beta_z \left(g - x \right) - v \right) + f\left(s \right) + \Delta u \\ \tau \dot{x} = v \end{cases}, \tag{6}$$

where Δu is the additional term, like $\gamma R v \phi \exp(-\beta\phi)$ in [33]. The parameters γ and β are known, and the variable ϕ and matrix R are observable. In our recent research, we considered the case that term Δu cannot be modelled but characterized by a set of discrete datasets and proposed the Constrained Dynamic Movement Primitives (C-DMP) [21] for skill generalization. However, the DMP-based model only addressed the constraints after learning skills, which may lead to unsatisfactory results in an environment with different constraints. For example, the demonstrations of obstacle avoidance maybe inherently contain limitations imposed by obstacles, and the learned skills may not generalize well to environments without obstacles or with more obstacles.

C. How to separate CI skills from demonstrations

Separating a CI skill from tool use demonstrations influenced by constraints is a core challenge to be resolved. We can specify the question from the mathematical perspective of DMP. DMP can represent an action learned from a demonstration or a series of actions. After adding the constrained conditions to (1), we can achieve (6), where Δu is the term influenced by the constraints. If Δu is well-modelled and the parameters in Δu are known, we can easily reduce Δu from (6) and achieve the CI skill expressed by (1), like the methods in [30]-[38].

However, during the demonstration process, the parameters in Δu are usually unknown or the constraints cannot be extracted with an exact model of Δu but are expressed by a series of discrete datasets like in [21]. Meanwhile, the constraints generated by the objects, tools and environment have been inherited in the demonstrations. Then the proposed methods in Section III are to separate the influence of Δu, achieve CI skills in (1) and adapt the learned results to new tasks with the new constraints generated from new objects and environments to realize skill transfer.

Here, we can use an example to illustrate the necessity of CI skill learning for tool-use cases. As shown in Fig. 2, we use a finger-shaped tool (blue) to push an object (red circle) forward

and leftward, generating trajectories of the object and the tool (Fig. 2(b)). The effective contact regions are shown in Fig. 2(c). Multiple trajectories are obtained by repeating object pushing action, as shown in Fig. 2(d), and DMPs are used to learn from these trajectories to achieve a red line in Fig. 2(e). We see that the learned trajectory, red line in Fig. 2(e), is smoother than the demonstrations and has fewer corners due to the state change of pushing actions, indicating that the learned skill diminishes the features of tool-use actions. Therefore, the learned results from demonstrations can not applied to new tools and tasks, because the generated trajectories may be not suitable for the given tools.

III. Multi-Tool Use Skill Learning and Transfer

In this section, we will carefully introduce how to realize the modules in Fig.1. Subsections A and B are for the O2 skill and the TF skill learning from human demonstrations, respectively. Subsection C is for skill generalization of both O2 skill TF skill and integration of the two skills to achieve a desired trajectory. Subsection D designs a controller to enable the robot to follow the trajectory to complete a new tool-use task.

A. Object operating (O2) skill learning

1) Model-based method

In the model-based method, the expression of Δu is known but with unknown parameters. Therefore, the term $f(s)$ cannot be achieved without knowledge of the exact expression of Δu.

The first is estimating a forcing function term integrating $f(s)$ and Δu by minimizing

$$\min\left(\sum_{j=1}^{N}\left(F_j^{Tar}-F(s,u)\right)^2\right). \tag{7}$$

where

$$F_j^{Tar}=\tau\dot{v}_j-\alpha_z\left(\beta_z\left(g-x_j\right)-v_j\right), \tag{8}$$

and

$$F(s,u):=f(s)+\Delta u. \tag{9}$$

Next, with knowing the general expression of Δu, such as $\Delta u=\gamma R v\phi\exp(-\beta\phi)$ in [33], we need to estimate parameters γ and β, and separate Δu from $F(s,u)$. The Δu is transformed by a function $T(*)$ and expressed as

$$T(\Delta u)=\boldsymbol{\theta}^T\boldsymbol{\Psi}+\sigma, \tag{10}$$

where σ is a noise term, $\boldsymbol{\theta}$ is an unknown parameter vector and $\boldsymbol{\Psi}$ is a state vector for identification. For example, Δu in [33] can be transformed as

$$\ln\left(\gamma R v\phi\exp(-\beta\phi)\right)=\boldsymbol{\theta}^T\boldsymbol{\Psi}+\ln\left(Rv_i\phi_i\right), \tag{11}$$

where $\boldsymbol{\theta}=\left[\ln(\gamma),\beta\right]^T$ and $\boldsymbol{\Psi}=\left[1,-\phi_i\right]^T$. Set $\hat{\boldsymbol{\theta}}(k)$ as the estimation of $\boldsymbol{\theta}$ in the kth calculation round, then the target value of $f(s)$ in (9) is calculated by $\hat{\boldsymbol{\theta}}(k)$ at the kth time as

$$f_j^{Tar}\big|_k=F_j^{Tar}-\Delta u(\hat{\boldsymbol{\theta}}(k)). \tag{12}$$

and $f(s)$ at the kth round is calculated by minimizing

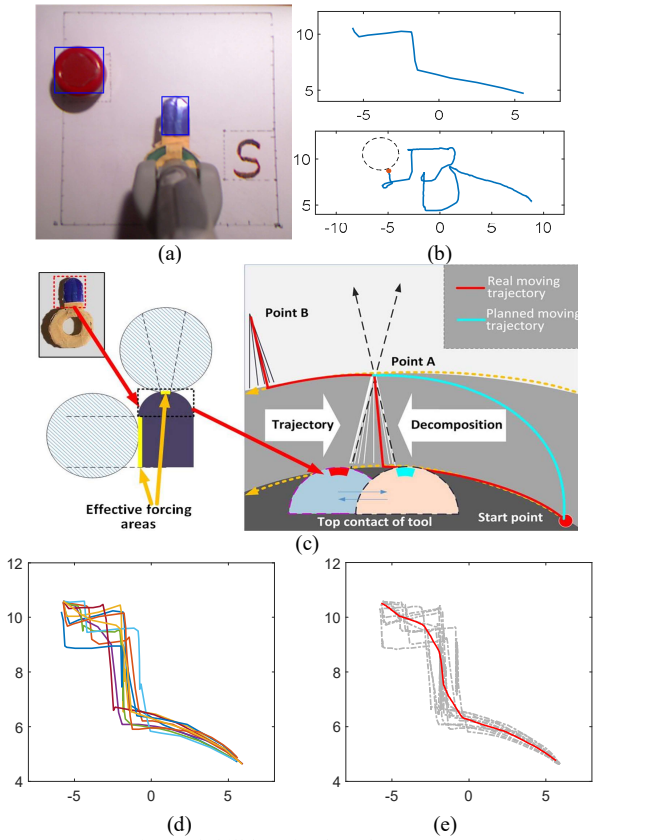


Fig. 2. Demonstration and skill learning based on DMP (a) Experiment setup (b) Trajectories of the tool's end and object center (c) The effective tool-using areas of the tool and the generated trajectory. (d) Object moving trajectories (e) Skill learning from demonstrations based on the standard DMP

$$\min\left(\sum_{j=1}^{N}\left(f_j^{Tar}\big|_k-f(s)_k\right)^2\right). \tag{13}$$

The vector $\hat{\boldsymbol{\theta}}(k+1)$ is updated based on the k*th* calculating results $\hat{\boldsymbol{\theta}}(k)$ and $f(s)_k$ as

$$\hat{\boldsymbol{\theta}}(k+1)=\hat{\boldsymbol{\theta}}(k)+\eta\mathbf{K}_n\left(T\left(F(s,u)-f(s)_k\right)-\hat{\boldsymbol{\theta}}(k)^T\boldsymbol{\Psi}\right) \tag{14}$$

where η is a constant updating factor and $\mathbf{K}_n$ is a gain matrix. It is obvious that if $\hat{\boldsymbol{\theta}}(K)$ converged at the calculation round $k=K$, then $f(s)\big|_{k=K}$ will approach to a stable results. Finally, we can get the stable results of $\hat{\boldsymbol{\theta}}(K)$ and $f(s)\big|_{k=K}$ as the final outcomes of O2 skill learning, the CI skill, is expressed as

$$\begin{cases}\tau\dot{v}=\alpha_z\left(\beta_z\left(g-x\right)-v\right)+f(s)\big|_{k=K}\\ \tau\dot{x}=v\end{cases} \tag{15}$$

and the constrained model is estimated as $\Delta u=T^+\left(\hat{\boldsymbol{\theta}}(K)^T\boldsymbol{\Psi}\right)$, where $T^+(*)$ represents the inverse calculation of $T(*)$. The calculation procedure is presented in Algorithm 1 in detail.

2) Constraint-based method.

Discontinuous limitations widely exist in the trajectory planning of autonomous vehicles [54] and robots [55]. Usually, the boundaries are obtained through digital maps [54] or the corresponding configuration [55]. For the O2 skill learning, we

Algorithm 1: Model-based object operating skill learning

Input: Human demonstration trajectory x_j

Begin:

1) Use (7) and (8) to learn a skill with constraints

2) Use (10) to transform the additional term Δu to a linear model

3) Initialize $\hat{\boldsymbol{\theta}}(0)$ in (12) and achieve $f_j^{Tar}|_0$

4) For k = 0 to K do

5) Use (13) to achieve $f(s)_k$

6) Use (14) to calculate $\hat{\boldsymbol{\theta}}(k+1)$ based on $\hat{\boldsymbol{\theta}}(k)$

7) Use (12) to update $f_j^{Tar}|_k$

8) **If** $\hat{\boldsymbol{\theta}}(k+1)-\hat{\boldsymbol{\theta}}(k)<\varepsilon$

9) Break whole loop and output $\hat{\boldsymbol{\theta}}(k+1)$ and $f(s)|_{k=K}$

10) **End**

11) End

Output: Model-based O2 skill x calculated by (15)

Algorithm 2: Constraint-based object operating skill learning

Input: Human demonstration trajectory x_j and constraints x^L, x^H

Begin:

1) Use (4) and (5) to learn a skill x with constraints

2) Use (18) and (19) to achieve the parameters k_H, k_L, and k

3) Taking the values of k_H, k_L, and k into (17) to achieve x^u

4) Use DMP (21) to generate a O2 skill x^u

Output: Constraint-based O2 skill x expressed by (21)

assume that the skill learned and expressed by (1) . Each point x is limited by constraints $x^L \le x \le x^H$. If we want to achieve a CI skill by removing the influence of constrained inequality, our solution is to find a transformation function from x to x^u, similar to the polar-like space analysis approach [69], to match the constrained space and the universal space with the unique mapping relationship.

To achieve this, we first learn a constraint skill from human demonstrations using (1). Next, a sigmoid-like function is built to convert x to x^u, thus removing constraints imposed by the boundaries:

$$x = x^L + \frac{\left(x^H - x^L\right)}{1+e^{-k\left(x^u-\left(k_H x^H + k_L x^L\right)\right)}}, \tag{16}$$

Here, k, k_H and k_L are non-zero constants, satisfying $0 < k_H, k_L < 1$ and $k_H + k_L = 1$. In (16), the relationship between x and x^u is unique. x^u can be achieved by the inverse calculation

$$x^u = -\frac{1}{k}\log\left(\frac{x^H - x}{x - x^L}\right) + k_H x^H + k_L x^L. \tag{17}$$

It is desirable that the start and end points will be not changed, that is $x^u = x$ at time $t = 0$ and $t = \infty$. Based on this condition, we can achieve the values of k_L, k_H and k:

Proposition 1 (The proof of Proposition 1 is presented in the Appendix A.): For the Sigmoid-like mapping function of x and x^u in (16) and (17), the essential condition of $x^u = x|_{t=0,\infty}$ is

$$\begin{cases} k_H = \dfrac{x_0^L g - x_g^L x_0}{\Delta x_1} + \dfrac{1}{k\Delta x_1}\Big[x_0^L \log\left(x_g^H - g\right) + \\ \quad x_g^L \log\left(x_0 - x_0^L\right) - x_0^L \log\left(g - x_g^L\right) - \\ \quad x_g^L \log\left(x_0^H - x_0\right)\Big] \\ k_L = \dfrac{x_0^H g - x_g^H x_0}{-\Delta x_1} - \dfrac{1}{k\Delta x_1}\Big[x_0^H \log\left(x_g^H - g\right) + \\ \quad x_g^H \log\left(x_0 - x_0^L\right) - x_0^H \log\left(g - x_g^L\right) - \\ \quad x_g^H \log\left(x_0^H - x_0\right)\Big] \end{cases}, \tag{18}$$

and

$$k = -\frac{1}{\Delta x_2}\Big[\Big(\log\left(x_g^H - g\right) - \log\left(g - x_g^L\right)\Big)\left(x_0^H - x_0^L\right) - \left(\log\left(x_0^H - x_0\right)\log\left(x_0 - x_0^L\right)\right)\left(x_g^H - x_g^L\right)\Big]. \tag{19}$$

where g and x_0 are the goal and start of x in (1), x_g^L, x_g^H, x_0^L and x_0^H are the boundary conditions of g and x_0, Δx_1 and Δx_2 are

$$\begin{aligned} \Delta x_1 &= \begin{cases} x_g^H x_0^L - x_0^H x_g^L, & if\ x_g^H x_0^L \ne x_0^H x_g^L \\ \sigma_1 & others \end{cases}, \\ \Delta x_2 &= \begin{cases} \Delta x_3 + \Delta x_1, & if\ \Delta x_3 + \Delta x_1 \ne 0 \\ \sigma_2 & others \end{cases}, \end{aligned} \tag{20}$$

where $\Delta x_3 = g\left(x_0^H - x_0^L\right) - x_0\left(x_g^H - x_g^L\right)$, and σ_1 and σ_2 are small enough values for ensuring Δx_1 and Δx_2 are not 0.

Final, we can use DMP in (1) again to learn and generalize a constraint-irrelevant O2 skill from x^u as

$$\begin{cases} \tau \dot{v}^u = \alpha_z\left(\beta_z\left(g - x^u\right) - v^u\right) + f^u(s) \\ \tau \dot{x}^u = v^u \end{cases}. \tag{21}$$

where $f^u(s)$ is achieved based on the new x^u and v^u. The constraint-based O2 skill learning is presented in Algorithm 2.

B. Tool Flipping (TF) skill learning

The TF skill is for the situation where the object keeps static and only the tool changes contact positions on the object. Figure 3 provides an overview of the TF skill. The pink tool changes contact position from left (point 2) to top (point 1) along a black solid trajectory and the contact position on the object is changed from the right to the bottom. This contact point switching path is influenced by the shapes of both the tool and the object. We can imagine that if the tool can be seen as a point, the possible moving trajectory is represented by the blue dashed line in Fig. 3. The extra displacement (a red arrow line in a zoomed figure) from Point 2 to Point 1 is caused by the tool's shape. Therefore, we proposed an idea of 'virtual object movement' to describe the displacement which is caused by the tool's shape. First, we can firstly learn from the tool's movement (black line) using DMP to achieve the expression

$$\begin{cases} \tau \dot{v}^{Tf} = \alpha_z\left(\beta_z\left(g^{Tf} - x^{Tf}\right) - v^{Tf}\right) + f^{Tf}(s) \\ \tau \dot{x}^{Tf} = v^{Tf} \end{cases}. \tag{22}$$

where x^{Tf} and v^{Tf} are the position and velocity of the tool's path, and $f^{Tf}(s)$ is the forcing function. Set x_0^{Tf} as the start of x^{Tf} and use the geometric relations at the start and end position shown in Fig.3, x^{Tf} is divided and expressed by two trajectories: the tool's movement x^t (blue dashed line that is also a CI TF skill) and a virtual object movement x^o (red dashed line):

$$x^{Tf} - x_0^{Tf} = k_1\left(x^t - x_0^{Tf}\right) + k_2\left(x^o - x_0^{Tf}\right), \quad (23)$$

where k_1 and k_2 are constant factors. Then, if the outline of the tool can be described using the following DMP function:

$$\begin{cases} \tau\dot{v}^o = \alpha_z\left(\beta_z\left(g^o - x^o\right) - v^o\right) + f^o(s) \\ \tau\dot{x}^o = v^o \end{cases}, \quad (24)$$

we can get the following proposition for the TF skill x^t.

Proposition 2: (The proof of Proposition 2 is presented in the Appendix B.): For a TF demonstration trajectory in (22), after generating a fixed virtual object's movement (24), the TF skill is learned and expressed in a DMP from as

$$\begin{cases} \tau\dot{v}^t = \alpha_z\left(\beta_z\left(g^t - x^t\right) - v^t\right) + f^{Tf}(s)/k_1 - k_2 f^o(s)/k_1 \\ \tau\dot{x}^t = v^t \end{cases}. \quad (25)$$

where k_1, k_2 satisfy

$$(I - k_1)(I + k_2)^{-1} = tr\left(\Delta x^t \left(\Delta x^o\right)^H\right) \quad (26)$$

where $\Delta x^t = g^o - x_0^{Tf}$ and $\Delta x^o = g^t - x_0^{Tf}$, and $tr(*)$ represents the trace calculation of a matrix.

Remark 1: As shown in (25), the TF skill has infinite solutions. We can see from the zoomed figure in Fig.3 that the constraints come from the start and end points in (23), selection of $f^o(s)$ in (24) and factors $k_i, i = 1,2$. The term k_2 reflects the influence of the virtual object movement x^o to TF skill learning process, but the trajectory x^o doesn't have a fixed shape. To avoid the conflict between the tool and objects, we can choose the tool's contour to generate x^o in (24) and $k_2 = 1$ to meet the minimum requirements for avoiding conflicts. Then we can get the unique k_1 and trajectory x^t :

$$\begin{cases} \tau\dot{v}^t = \alpha_z\left(\beta_z\left(g^t - x^t\right) - v^t\right) + \left(f^{Tf}(s) - f^o(s)\right)/k_1 \\ \tau\dot{x}^t = v^t \\ k_1 = I - 2tr\left(\Delta x^t \left(\Delta x^o\right)^H\right) \end{cases} \quad (27)$$

We can reduce conflicts between the object and the tool by enlarging k_2 in skill learning and generalization process.

C. Skill generalization

The model-based O2 skill can be generalized by the methods in [30]-[38] based on the model with the estimated parameters $\hat{\boldsymbol{\theta}}(K)$. The generalization of TF skill is also easy to realize: first generalizing object virtual movement x^o using (24) and the CI movement of the tool using (25) incorporating the new flipping

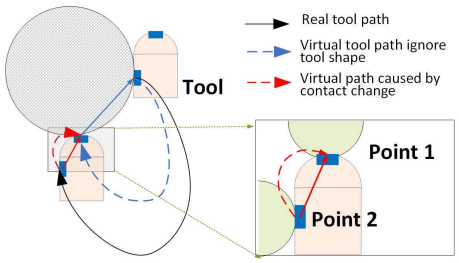


Fig. 3. Decomposation of tool flipping (TF) skill

contact points separately. After acquiring x^t and x^o, the new TF movement of the tool is calculated by

$$x^{Tf} = k_1 x^t + k_2 x^o + \left(1 - k_2 - k_1\right) x_0^{Tf}. \quad (28)$$

where x_0^{Tf} represents the initial contact point of TF actions. The primary focus of our contribution is on the generalization of constraint-based O2 skills, building upon our earlier research [21], which necessitates prior knowledge of constraints in a new situation and the reference trajectory x^c. In this section, we enhance this method and propose Constrained-DMP(C-DMP) lite. First, we assume that the generalized skill is expressed as

$$\begin{cases} \tau\dot{v}^N = \alpha_z\left(\beta_z\left(g - x^N\right) - v^N\right) + f^u(s) + \Delta u^c \\ \tau\dot{x}^N = v^N \end{cases}, \quad (29)$$

where Δu^c is an additional constrained term. Similar to (6), x^N and $\dot{x}^N$ represent new trajectory and velocity. x^{NL} and x^{NH} are constraints of x^N, having the same conditions to the constraint-based learning process: $x^N(0) = x^u(0)$ and $x^N(\infty) = x^u(\infty)$. x^u is the generalized CI trajectory. Define $\underline{h}_i = x^{NL} - x^u$, $\overline{h}_i = x^{NH} - x^u$, it is desired that x^N in (29) bounded with

$$\underline{h}_i < x^N - x^u < \overline{h}_i. \quad (30)$$

We build a novel Lyapunov function candidate V^c :

$$V^c = \frac{1}{2}\log\frac{\left(\overline{h}_i - \underline{h}_i\right)^2}{4\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)} + \frac{1}{2}k_n\tau z_2^2. \quad (31)$$

where $z_1 = x^N - x^u$, $z_2 = v^N - \lambda$, and $\lambda = \tau\dot{x}^u - z_1 + \left(\overline{h}_i + \underline{h}_i\right)/2$ is a virtual term for ensuring system stabilization, and k_n is a positive number. Then skill generalization condition for (29) is concluded in Theorem 1 as:

Theorem 1: (The proof of Theorem 1 is presented in Appendix C.) For the DMP function in (29) with the Lyapunov function candidate V^c in (31), then the sufficient condition for $\dot{V}^c < 0$, enabling x^N to satisfy (30), is

$$\Delta u^c = \alpha_z\beta_z z_1 + \left(\alpha_z\tau - \tau\right)\dot{z}_1 - \left(k_z + \frac{1}{\tau\left(\overline{h}_i - \underline{h}_i\right)^2}\right)\frac{z_2}{k_n}, \quad (32)$$

where $k_z > 0$ is constant factor.

Remark 2: Compared with [21], the expression in (32) is more concise, requiring fewer manual configurations and featuring a shorter stability proof. Notably, there is no requirement for the

reference point x^c, as calculation of Δu^c can be performed by (32) directly. Consequently, the C-DMP lite reduces calculation complexity by at least 30%, thus eliminating time-consuming calculations of x^c, k_n and $\dot{\alpha}_2$. Furthermore, compared to other improved DMP methods mentioned in [39], this approach does not depend on acceleration information, which is often subject to inaccurate measurements.

D. Controller design

After separately generalizing the O2 skill and the TF skill, a complete trajectory is established for manipulation using a kind of skill or connecting the two skills to represent the desired path x^d for robot, and we can calculate robot desired joint angle of q^d using the inverse kinematic model.

Therefore, the final step involves controlling robots to follow the desired trajectory. In general, a robot system is expressed as

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G=J^T(q)F_e-\tau, \qquad (33)$$

where q, $\dot{q}$ and $\ddot{q}$ represent angle, velocity and acceleration of robot joints, $M(q)$ and $C(q,\dot{q})$ are the inertia matrix and the centripetal and Coriolis matrix, G is the gravitational torque, $J(q)$ is the Jacobian matrix and F_e represents the environmental force from interaction process with bounded values. τ is the control torque to be developed as

$$\tau = M(q)\left(\ddot{q}^d+k_e\dot{e}\right)+C(q,\dot{q})\left(\dot{q}^d+k_e e\right)+G+\xi r_e, \qquad (34)$$

where q^d and $\dot{q}^d$ are the desired angle and angle velocity of the joints, which are calculated by the trajectory achieved by skill learning methods in Section III. B and generalization results in Section III.C. Set $e=q^d-q$ is the joint tracking error , the term $r_e=\dot{e}+k_e e$, where k_e is a constant factor and ξ is the constant impedance factor. Following the previous research [28], [29], and [59], the controller stability is easy to be proved by building Lyapunov bounded stable conditions.

IV. Experiments

This section conducts several experiments to validate the effectiveness of the proposed framework. The first experiment is conducted with a desktop Omni joystick and two self-made tools, with accurate kinematic and dynamics models of robots and the tools for error quantification. The second experiment employs a Franka robot and several real-world tools to verify generalizability of the proposed framework. The organization of this Section is: Sections IV.A and B pertain to the first experiment and its comparative experiment, while Section IV. C focuses on the second experiment.

A. Experiment 1: a pusher-slider and obstacle avoidance task

Pusher-slider and obstacle avoidance experiments are typical for verifying tool-use skill ability. In the experiment, we verify both kills simultaneously. Fig. 4 illustrates the experimental setup, comprising a Kinect, a Joystick, two tools (coloured blue), and a red round object. The tools are attached to the end of the Joystick, which serves both as a demonstrator to record the positions of the robot end and as an actuator to execute planned actions. The experiment follows the procedure outlined

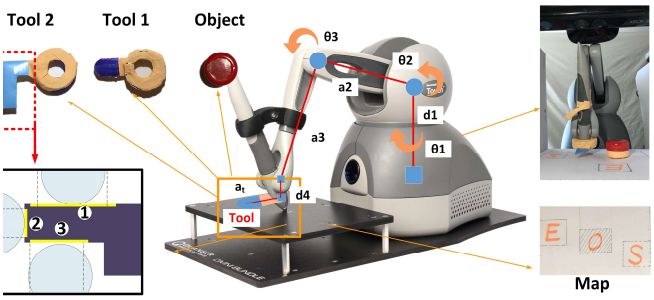


Fig. 4. Experimental setup.

TABLE 1 Acquired information of the tools, objects, obstacles and maps

	Pre-acquired Information
Map	Size: 180mm*180mm, Demonstration: Start at [6,5.5] and end at [-6,13]; Generalization: Start at [6,8] and end at [-5.5,11];
Tools	**Tool 1**: Length = 30mm; Width = 12mm; Radius of contact head =6mm; Head contact angles = [-0.08 rad, 0.08 rad]. Left contact length = 10mm; **Tool 2:** Length = 40mm; Width = 40mm; Contact length of 1st surface =15 mm, Contact length of 2nd surface =10 mm, Contact length of 3rd surface=20 mm in Fig. 8;
Object	Radius =12 mm;
Obstacle in map	Rectangular obstacle: width=30mm, height =40mm, center position is [0 10];

in Fig.1, which consists of several steps for pushing the object from the starting region 'S' to the end region 'E.' The shaded area 'O' represents an obstacle that must be avoided.

We developed two tools for manipulation. Tool 1 has a stick shape with a round head, while Tool 2 has an L shape with three surfaces (areas marked as 1, 2 and 3 in Fig. 4) for generating pushing actions in three directions. These tools are considered as typical examples of the pull-from-tube tasks described in [8] and the shape primitives of the tool in the simulation presented in [50]. Effective contact regions are set manually or recognized through demonstrations. During the experiments, movements of both tools and objects are recognized by the camera and the robot movements are recorded by the Joystick. The geometric information of tools, objects and obstacles is known beforehand. This allows us to obtain information for the skill generalization process. The detailed information about objects, tools, obstacles, and maps used in the experiments is provided in Table 1.

The constraints are derived from the effective regions of the tool, obstacles, and the actuator's effective working space, with the following conditions: the tool's operational directions are perpendicular to the contact surfaces, and the object doesn't slip on the tool's surface. We then utilize human demonstration data, as shown in Fig. 2(b) and divide, resample, and align them to create a dataset for learning the O2 and TF skills. The processed demonstrations are represented as the grey lines in Fig. 2(e) for the O2 skill and in Fig. 6(c) and (d) for the TF skill.

1) O2 skill learning and generalization

Learning O2 skills is subject to the constraints imposed by the tool's usages and robot's workspace. To account for these constraints, we define x^T as the contact point positions on the tool, x^A as the actuator's end position, and x^O as the object center position with the relationship $x^T=T_A^T x^A$, $x^O=T_T^O x^T$,

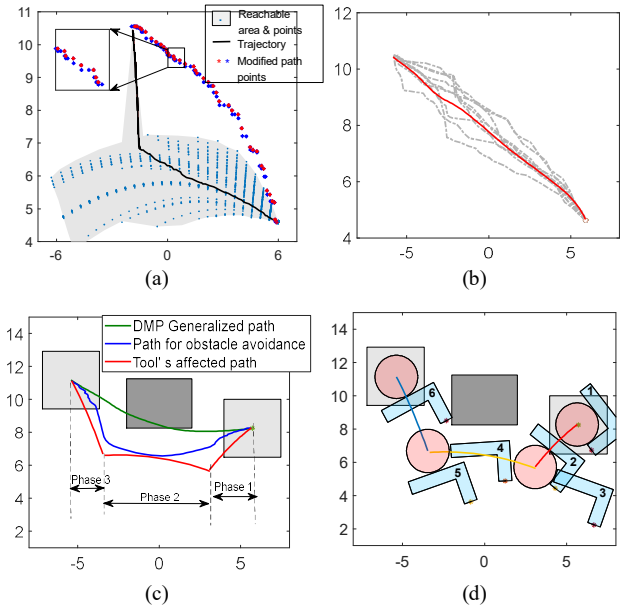


Fig.5. Skill learning and generalization of O2 skill for a pushing and obstacle avoiding task (a) constraints and unconstrainted trajectory for a fragment in a demonstration; (b) unconstrainted trajectories and skills learned from them; (c) generalized skills (the skill is generalized three times: the green line shows the generalization of the learned CI skill, the blue line is the generalization for obstacle avoidance based on the green line, and the red line is a generalized skill considering the tool and the acutator's constraints and based on the blue-trajectory; (d) illustraion of using a L-shape tool to push the object to the target (O2 skill part)

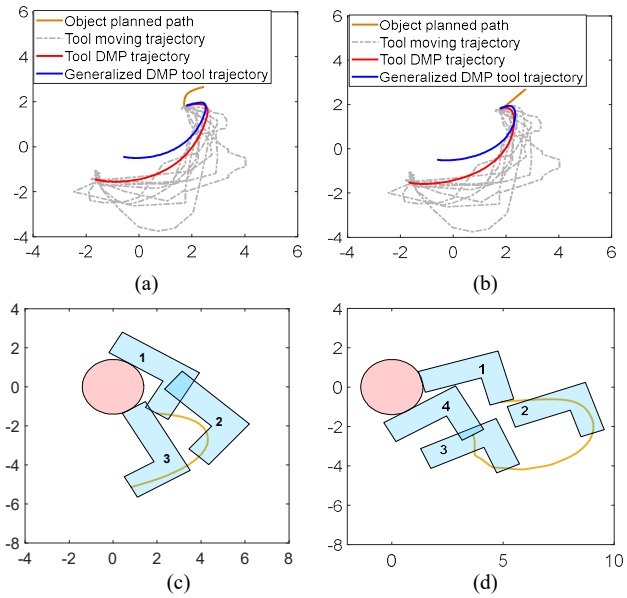


Fig. 6. Constraints and the recovered demonstrations and the skill learned from the processed data. (a) TF skill learning for a arched virtual object movement; (b) TF skill learning for a line virtual object movement; (c-d) TF skill generalizations for different operational requirements by using a new tool

where T_A^T and T_T^O are the transforming matrices. The reachable areas of tools from the start to the end for each fragment are presented as the grey shaded areas with blue dots in Fig.5 (a), while the black line represents the tool's movement within the constrained region. These constraints cannot be modelled as an exact function but can be limited to the boundaries of the grey areas.

Using (17), we can transfer the trajectories of two sections into a universal trajectory mapping to the entire region between two points, presented by the red and blue dot lines. Other object trajectories are transformed similarly, resulting in a group of CI actions, as shown by the grey lines in Fig.5 (b). A CI skill is then learned from these CI actions using the standard DMP. The model-based method is not applicable because the constraints are dispersed and cannot be modelled.

Different constraints lead to different generalized O2 skills. As shown in Fig.5 (c), the green line represents the trajectory generalized from the learned O2 skill without any constraints. To avoid conflicts between obstacles and objects, we introduce a 1.5cm width 'safe margin', as used in [57], to achieve the blue line shown in Fig. 5(c) using Constrained-DMP lite. In our prior work [21] and the DMP-based obstacle avoidance methods, e.g., [33], the influence of tool contact regions to the trajectory is rarely considered. Thus, the blue line is the final trajectory for robot manipulation. Here, we add new constraints from tool-use cases where the three areas in Fig.4 are selected as the potential contact areas. The generalized object's movement is shown as red lines in Fig. 5(c). Due to the frequently changing contact regions, the object's motions are completed in three segments, within the robot's movement ranges. Especially, in phase 2, the generalized trajectory suffers from the constraints of the tool's reachability and 'safe margins' to avoid conflicts with obstacles. Using the transformation function $x^O T_O^A T_A^T = x^T$, the movements of the L-shape tool are calculated and marked in Fig.5 (d) in sequence to push forward and leftward the object from the start to the destination.

2) TF skill learning and generalization

In Fig.6 (a) and (b), we compare the results of learning from the same demonstrations using different tool shapes: the red line represents skills learned from the original, the orange line represents the tool's outline x^o and the blue line represents a TF skill x^t achieved by (23). It is evident that the trajectory in Fig. 6 (a) significantly derivates from the starting point to avoid conflicts compared to Fig. 6 (b), even when the start and end points are the same.

The TF skill can connect O2 actions between two adjacent phases, as illustrated steps 2 and 3, and steps 4 and 5 in Fig. 5 (b). Fig. 6 (c) and (d) illustrate two examples that we need to know the initial and the final contact positions and poses for the objects and tools. For example, we set the position of the object is (0,0) and the radium is 1.5, the initial and final contact angles are $[1/3rad, -1/3rad]$ and $[1/12\,rad, -1/3rad]$. The generalized trajectories are presented in Fig. 6 (c) and (d) and we mark the flipping processing using numbers. We can see that both objects and tools change contact positions throughout the TF process to enable the object to be pushed along the desired direction in the following O2 actions.

3) Skills integration and application

The final step involves connecting the generalized O2 skills and TF skills to generate a complete trajectory for the process. As shown in Fig. 7 (a), the purple lines represent the planned

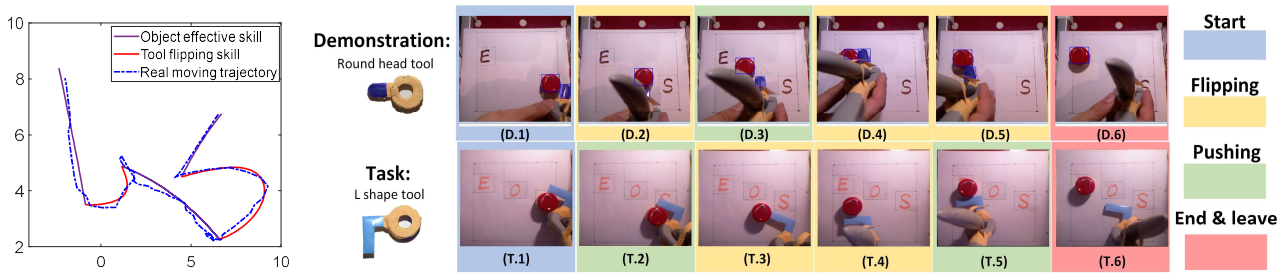


Fig. 7. The whole-process skill generealization and experimental certification for goal reaching task by integrating the O2 and TF skills; (a) Generalized and real measured trajectories; (b) Demonstrations using the old tool and experimental certifications for generalized skills based on the new tool.

robot end trajectories achieved in O2 skill generalization (as depicted in Fig. 5(d)), and the red lines represent the TF skill generalization results used to connect these O2 skills based on the end positions, such as (-1, 3.5) achieved in Fig. 5(d) for the step 5, and its pushing directions of the O2 trajectories, similar to the results in Fig.6 (c) and (d). By using the controller (34) and setting the control parameters as $k_e = 3$ and $\xi = 10$, the joystick with Tool 2 is effectively controlled to push and pan the object, enabling it to follow the desired way, circumvent obstacles, and reach the destination successfully. The final real trajectory is shown as the blue dashed line in Fig. 7 (a) and the realization process is illustrated in Fig. 7 (b).

Remark 3: Upon achieving the planned trajectory, the focus shifts to controlling the robot to follow the desired path. Our previous research dealt with several cases that combined DMP and adaptive impedance control [16], [28]-[29] as well as neural network (NN)-based control [58] to ensure stable interaction with the objects and the environment, taking into consideration uncertain dynamics and external disturbances. In this paper, we develop an impedance controller to minimize position tracking errors. Furthermore, the proposed TF skill allows for real-time modifications of contact points on the same tool. Therefore, in case of large errors occurring along the way, we can adjust the contact points in real-time to achieve trajectory modifications.

B. Comparison with other methods

To evaluate the performance of the proposed framework and method, we compared the O2 skill and TF skill learning results with the baseline methods separately. For DMP-based O2 skills, we compare the results in Fig.5(d) with the trajectory achieved by the two-level RRT* method [70]. We utilize the same map and obstacle depicted in Fig. 4 and the parameters in RRT* are as follows: a displacement of 0.2 cm to the goal, a maximum step length of 0.2 cm, the neighboring node radius of 1cm, and the number of the closest vertex (node) extensions set to 7. The maximum number of nodes is set to 1400 and the satisfaction condition returned true only for configurations within 0.2 cm displacement to the goal.

The simulation results about the object motion trajectory are shown in Fig.8 (a), where the thick highlighted blue nodal lines represent the final trajectory from the starting point to the target region, and the thin dark blue short lines represent the extended trajectories. We can observe that the end of the trajectory does not coincide with the destination, even if the target position is known beforehand using the RRT* method. However, within the DMP, a trajectory is generated based on the start and end points, and the forcing function in (1) decreases to zero at the end, ensuring that the trajectory stops at the target.

Based on the results in Fig.8 (a), we further make a tool-level movement planning. If we don't consider the constraints of the pushing directions and working regions of the tool, any contact point can be selected to push the object along the planned trajectory to the target, as shown in Fig.8 (b). However, there exist some constraints, such as the location of the joystick base, reachable regions, and limited DOFs of the actuator (only three joints are driven by the motors of the joystick), the tool should select several available contact points to push the object from one node to the next generated by RRT*. As illustrated in Fig. 8 (c), the tool changes contact points (22 points) throughout the path in three phases colored purple, green, and orange in Fig. 8 (d). It is notable that the L-shaped tool changes contact surfaces only twice, from the bottom side to the left side and then from the left side to the top side, resembling the DMP-based method results in Fig.5(d).

The difference lies in the fact that the RRT*-based method always requires changing contact points to push the object in the planned directions, while the DMP-based O2 skill learning only needs three contact points to push the object to the target with fewer calculations. Additionally, the RRT* based method does not fully consider the multiple constraints of the tool and environment during object-level planning, which may result in the generated tool positions that are not reachable under these constraints. In contrast, the proposed DMP-based method fully considers these constraints, generating an object trajectory from start to destination that can't be generalized to a new case by the RRT* based tool-use action. However, DMP can be combined with RRT* to implement path rescheduling as shown in [71] allowing for human-like responses through the use of human demonstrations.

We compare the computation complexity of the two methods. Following the depictions in [60], the computational complexity of DMPs depends on the regression calculation to minimize the weighted quadratic error in (4). Generally, we can use Locally Weighted Regression(LWR) and Locally Weighted Projection Regression (LWPR) to perform the regression, but the two methods require different computational complexities. For LWR, it is polynomial $O(n^2)$, and for LWPR, it is linear $O(n)$, where n represents the size of the data set. The proposed framework comprises several improved DMPs, where the main calculation burden still lies in the regression calculation. Therefore, the number of calculations increases linearly and the

computation complexity ranges between $l\mathrm{O}(n)$ and $l\mathrm{O}(n^2)$, where l depends on the division of tool-use phases and iteration number for the identification. Moreover, the computational complexity or time is also determined by the number of the basic function ψ_i in (2) [60]. FFT* method and its improved methods exhibits a computation complexity of $\mathrm{O}(n\log n)$ [61], a value between $\mathrm{O}(n)$ and $\mathrm{O}(n^2)$, where n represents the number of samples. Therefore, the computational complexities of the two algorithms are at the similar level. For the cases with a large number of samples, it is suggested to use LWPR to achieve more efficient calculations. According to [60], the average computational time of DMPs ranges from 10^{-4} s to 10^{-3} s. The proposed framework increases the computational complexity linearly, and the time is also linearly increased from 10^{-3} s to 10^{-1} s, as verified through simulation using MATLAB. A detailed summary of the comparisons is presented in Table 2.

We further compare the results of the classic DMPs for tool flipping actions, as presented in Fig. 9 (a) and (b), with those from TF skills, which are shown in Fig. 6 (c) and (d) under the same conditions. Our method of decomposing and recombining the learned DMP skills for tool flipping has two advantages. Firstly, it enables the tool to change contact points and directions during the tool flipping process and final states simultaneously. While the classic DMP-based approach results in conflicts and cannot be used to realize this purpose, as indicated in Fig. 9. It is due to that the shape of the tool and contacting point changes on the tool are not considered such that the object may be knocked out by the tool. Secondly, by choosing $k_2 = 1$, the range of robot motion is smaller based on the proposed approach. For example, the right range of the trajectory in Fig.9 (a) is approximately 8 cm, while it is only about 5 cm in Fig.6 (c).

C. Applications on robot arms and real-world tools

The proposed methods are executed using a Franka robot and real-world tools, such as cups and knives, to perform pouring and cutting tasks. The primary difference between real-world tools and 3D-printed tools is that the geometry of the real tools is not accurately modelled. Therefore, human demonstrators are required to indicate the functions and effective working regions of the tools through human demonstrations. Simultaneously, the demonstration trajectories are recorded for skill learning and generalization. Fig. 10 illustrates the processes of robot tool-use manipulations for cutting and pouring with different objects. The human demonstrator only demonstrates once, and the rest of the actions are performed by the robot autonomously. Fig. 11 illustrates the results of interacting with different objects.

In Fig. 10(a), we can observe that the generalized actions can adapt to different object requirements, such as slicing a banana into three parts and dealing with objects of different shapes and sizes, such as biscuits, grapes, strawberries, and meatballs. We can compare the incision made, when cutting the banana using the tool-use method (Fig.11(b)) with that using general skills learned from demonstrations (Fig. 14(a)). Since the sections cut by the tools are chosen to constrain the robot's motions, the

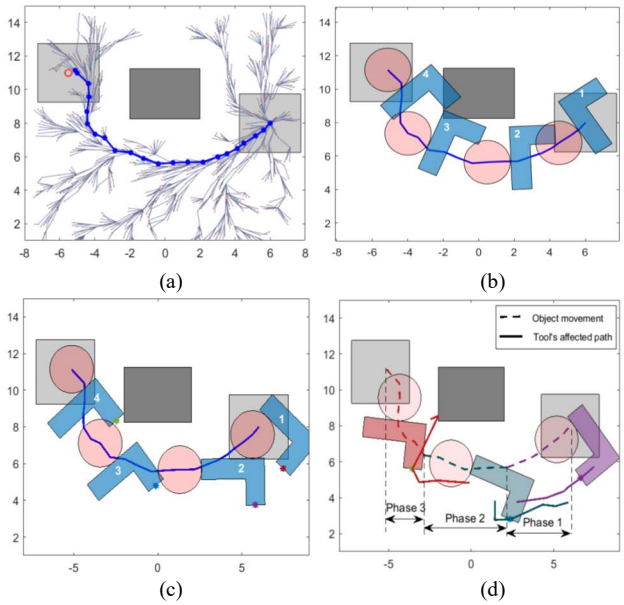


Fig.8. Two-level tool-use skill trajectory planning using RRT* method (a) object movements (blue thick line with 22 trajectory nodes is the final object movement and dark thin line segments are extended trajectories. (b) pushing motions without limitations of the tool's effective using regions and directions; (c) tool's pushing motions without constraints of tool's effective contact directions and regions; (d) three phases of tool's motions from the start to the destination(the purple tool with dash and solid lines in the same color represents the tool pushing states, object's motions and the tool's motions in the 1st phase, the green tool with green lines represent those in phase 2 and the orange tool with orange lines respresent results in phase 3)

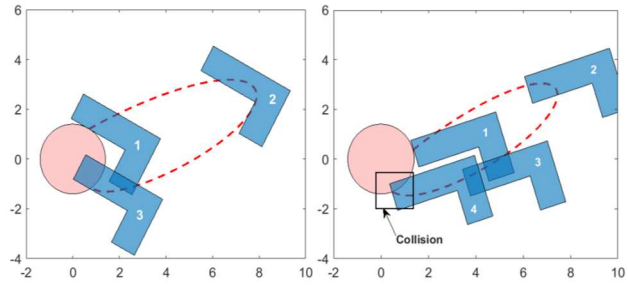
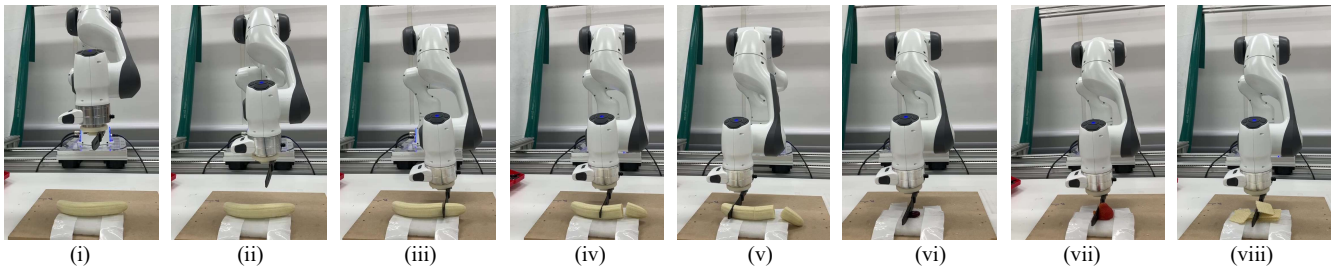


Fig. 9. TF skill learning and generalization using a new tool to realize tool flipping. (a-b) The comparative results of using the general DMP technology to those in Fig. 6 (c) and (d)

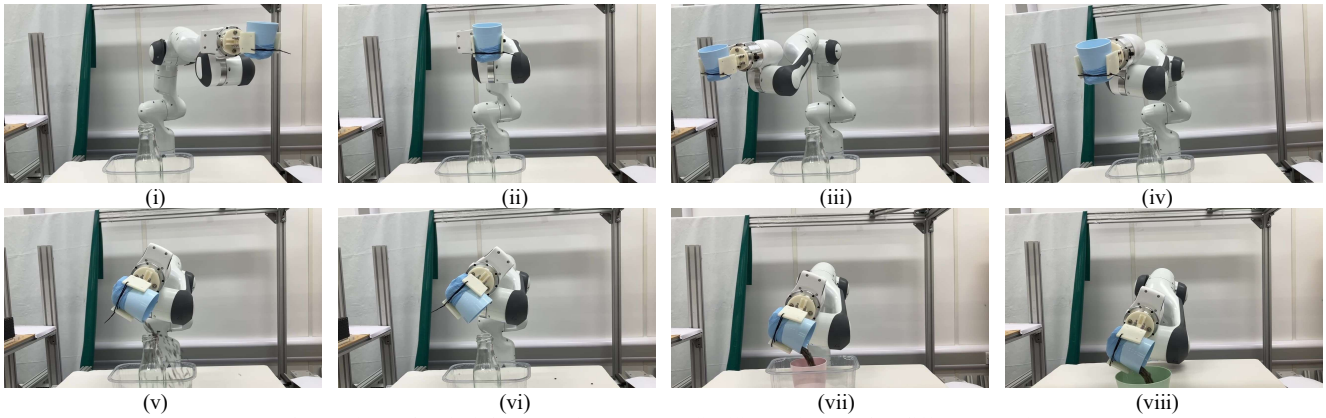
TABLE 2 Comparison of the proposed method with Two-Level RRT* tool use trajectory:

	Proposed DMP -based method	Two-level RRT*
Levels of tool action planning	2	2
Phases of tool movement	3	3
Number of trajectory points/tool trajectory rescheduling times	4	22
Continuous tool trajectory within each phase	Yes	No
Feasibility of reaching the goals	Yes	May not
If a generated position is unreachable by the tool	No	May yes
Skill generalization ability	Yes	No
Incorporation of human demonstrations	Yes	No
Computational complexity	$lO(n) \sim lO(n^2)$	$O(n\log n)$
Computational time	10^{-3} s to 10^{-1} s	10^{-3} s to 10^{-1} s



(i) (ii) (iii) (iv) (v) (vi) (vii) (viii)

(a) Object cutting tasks (i-v) banana cutting process (vi-viii) cutting other different objects(grape, strawberry and wafer biscuits)



(i) (ii) (iii) (iv)

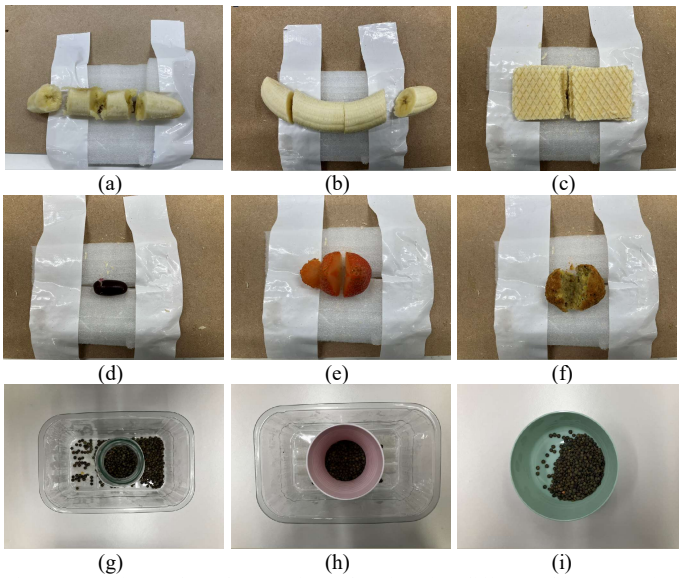
(v) (vi) (vii) (viii)

(b) Pouring tasks (i-vi) bean pouring process (vii, viii) pouring to other different containers

Fig.10. Experiments taken on real robot and tools.

incisions are much smoother, and the actions are adaptable to meet various object-cutting demands (Fig. 11(c) to (Fig. 11(f)).

For the pouring task, actions are generated encompassing two sub-actions: approaching the desired position and pouring the objects into the particulate matter into different containers. These actions are governed by constraints such as the shape of the pouring cup, the pose, position, and shape of the object containers, as well as the requirement that the relative distance of pouring edges on the two containers must be sufficiently close to ensure a successful pour. In the case of the second sub-action, consideration needs to be given to avoiding conflicts within a confined space. Therefore, we utilize the O2 skill to achieve goal approaching and the TF skill to ensure pouring accuracy. As depicted in Fig. 10(b), the robot learns pouring skills from human demonstrations and successfully pours the beans into the target cups. Figs. 11 (g) to (i) illustrate that only for the tall bottle with a narrow mouth, some beans are spilt, while the other two wide-mouth containers collect all the beans, thereby verifying the effectiveness of the action execution.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig.11. Experimental results. (a)-(f) results of cutting different objects; (g)-(i) results of pouring beans to different containers.

Through the above experiments, we know that the O2 and TF skills can be used independently and are compatible with other DMP methods. For example, the pouring experiment is taken by integrating the TF skill and O2 skill to complete the whole action, while the cutting experiment is only based on the O2 skill. Compared to other DMP methods, the learned CI skills can eliminate the influence of constraints from the special tools that occurs during the human demonstration process and enable the skill to have stronger generalizability.

V. Discussion

A. *Autonomy of skill learning and generalization*

The effective region and function recognition of tools in tool-use applications is the precursor to tool manipulation [52], [53], [56]. The DMP-based skill learning and generalization methods proposed in this paper belong to action planning of non-causal tool-use tasks. Therefore, the methods are taken after acquiring prior knowledge of tools, objects, and environments by utilizing analytical models and deep learning algorithms to identify the features [13],[52] and key points [51] and generate parametric constraints from demonstrations [46]-[48]. Then these methods, combined with the proposed DMP-based tool-use skill learning framework and controllers, like (34), can lead to autonomy in skill learning, generalization and action execution to satisfy the

assumptions or manipulation requirements (as those introduced in Section IV).

There are also some research integrated modeling, and skill learning and control of robots to generate actions and policies directly. For example, Seita *et. al* utilized 3D dense point flow prediction [63] and the learned perception model [72] to predict the tool and object state for further manipulation. Pertsch *et al.* [73] and Wang *et al.* [74] utilized reinforcement learning (RL) for robot interaction with environments and completing block stacking tasks. The difference is these methods need to build observation-action pairs, and the actions are generated by the observations (states) or networks. It needs to learn the policy through demonstrations and setting some parameters with some calculation time. It is obvious that the methods can generate the actions directly without the need for the separated steps for tool recognition and tool-use planning. However, the research did not realize skill transfer from one tool to another and the actions are not continuous and expressed in a formulation.

Comparatively, DMPs can conclude and express a skill from multiple similar demonstrations. The calculation of DMP could filter trajectory deviations and achieve a smooth trajectory from the start to the end. DMP is also a one-shot learning method that does not need a long time for policy resignation. Meanwhile, manual settings based on human experience and requirements are still necessary, such as the "safe margin" width and control parameters. These requirements are easy to present as the constraints to add to the learned trajectory without the need of retraining the network and policies. However, the DMP-based method is still a non-causal tool-use, where the pre-knowledge about the tools, objects and the environment are so necessary to generate constraints and trajectories, which is not flexible and cannot be updated through training in the simulation and real-world environments.

B. Incremental calculation complexity

Compared with Reinforcement Learning and other network-based learning methods, the calculation complexity of the DMP -based method is much smaller such that the results are learned and achieved within a short period or even real-time execution. Therefore, we only compare the incremental complexity of the proposed method with the classic DMP method.

Regarding the model-based learning method of the O2 skills, incremental computations are spent on the estimation of $\hat{\boldsymbol{\theta}}(k)$. Therefore, it requires several iterations of DMP calculations to achieve $\hat{\boldsymbol{\theta}}(K)$, and then the calculation complexity increases K times to obtain Δu in (9) for a new skill. Generalization does not need extra computations since the model for Δu is fixed. Fortunately, the identification process converges within a few steps, which can minimize the calculation time required. In the case of the constraint-based O2 skill, after calculating the parameters k_H, k_L and k using (18) and (19), additional computations are performed to extract CI actions via (17). The actions are learned using DMP, and no extra computations are required. The improved skill generalization method presented in [33], known as constrained DMP lite, can reduce complexity compared to the methods in [33] , as it eliminates the need for additional calculations of x^c and $\dot{\alpha}_2$. In the context of TF skill learning, the original TF actions are divided and expressed using two DMP functions, resulting in three DMPs. Therefore, the computations for skill learning and generalization are tripled. In the overall framework, human demonstrations are separated into several O2 and TF actions, which are learned and generalized separately and achieve an action chain for a new tool-use manipulation task. So the complexity combines the calculations of the two skills. The computational complexity is higher than that of the original DMP and other improved multi-phase DMPs e.g., [29]. However, the proposed method can effectively avoid conflicts and ensure that the object can follow the desired trajectory when using special tools to move within the robot's reachable areas.

VI. Conclusion

This paper presents research on robot tool-use skill learning and transfer based on DMP to enable robots to adapt tool-use skills from one tool to another in a new task. We propose a new framework comprising two skills: the O2 skill and the TF skill, along with several technical innovations based on DMP relating to action decomposition and combination. The O2 skill can be achieved in two ways, depending on whether the constraints can be modeled or expressed with discrete datasets. Additionally, we improved our previous work [21] by reducing over one-third of the computations required for skill generalization. The TF skill ensures the continuity for generalizations of O2 skills.

We perform several simulations and experiments to verify the effectiveness of both independent and joint applications of the O2 skills and TF skills. The results illustrate that, compared to the two-level RRT* tool-use trajectory planning, the O2 skills achieve a continuous and accurate trajectory with fewer rescheduling times. The TF skill can avoid conflicts between tools and the object during the moving period and at the contact moment. Experiments conducted on the Franka robot platforms focus on the verification of the generalizability of the proposed framework and independent methods.

Although the improved methods are only based on DMPs, the framework is not limited to this method. Some recently proposed methods, such as kernelized movement primitive [49], can also be incorporated into the framework as well, to reduce calculation complexity to some extent.

Appendix

A. Proof of Proposition 1

In the trajectory transformation, it is desired that start and end points should not be changed, namely $x^u = x\big|_{t=0,\infty}$, then we can create the following condition:

$$\begin{cases} g = -\dfrac{1}{k}\log\left(\dfrac{x_g^H - g}{g - x_g^L}\right) + k_H x_g^H + k_L x_g^L \\ x_0 = -\dfrac{1}{k}\log\left(\dfrac{x_0^H - x_0}{x_0 - x_0^L}\right) + k_H x_0^H + k_L x_0^L \end{cases}. \tag{35}$$

By setting $x_g^H x_0^L \neq x_0^H x_g^L$, k_H and k_L can be calculated by:

$$\begin{cases} k_H = \dfrac{x_0^L g - x_g^L x_0}{x_g^H x_0^L - x_0^H x_g^L} + \dfrac{1}{k\left(x_g^H x_0^L - x_0^H x_g^L\right)}\Big[x_0^L \log\left(x_g^H - g\right) + \\ \quad x_g^L \log\left(x_0 - x_0^L\right) - x_0^L \log\left(g - x_g^L\right) - x_g^L \log\left(x_0^H - x_0\right)\Big] \\ k_L = \dfrac{x_0^H g - x_g^H x_0}{x_0^H x_g^L - x_g^H x_0^L} + \dfrac{1}{k\left(x_0^H x_g^L - x_g^H x_0^L\right)}\Big[x_0^H \log\left(x_g^H - g\right) + \\ \quad x_g^H \log\left(x_0 - x_0^L\right) - x_0^H \log\left(g - x_g^L\right) - x_g^H \log\left(x_0^H - x_0\right)\Big] \end{cases} . \tag{36}$$

According to $k_H + k_L = 1$, we can get

$$\frac{1}{k}\left[\left(x_0^L - x_0^H\right)\log\left(\frac{x_g^H - g}{g - x_g^L}\right) - \left(x_g^L - x_g^H\right)\log\left(\frac{x_0^H - x_0}{x_0 - x_0^L}\right)\right] + \left(x_0^L - x_0^H\right)g - \left(x_g^L - x_g^H\right)x_0 = x_g^H x_0^L - x_0^H x_g^L . \tag{37}$$

Then, if $g\left(x_0^H - x_0^L\right) - x_0\left(x_g^H - x_g^L\right) + x_g^H x_0^L - x_0^H x_g^L \neq 0$ is satisfied, the general expression of k is

$$k = -\frac{\log\left(\dfrac{x_g^H - g}{g - x_g^L}\right)\left(x_0^H - x_0^L\right) - \log\left(\dfrac{x_0^H - x_0}{x_0 - x_0^L}\right)\left(x_g^H - x_g^L\right)}{g\left(x_0^H - x_0^L\right) - x_0\left(x_g^H - x_g^L\right) + x_g^H x_0^L - x_0^H x_g^L} . \tag{38}$$

B. Proof of Proposition 2

Following (23) and setting g^t and g^o as the end position of x^t and x^o, we have

$$\begin{cases} x^{Tf} - x_0^{Tf} = k_1\left(x^t - x_0^{Tf}\right) + k_2\left(x^o - x_0^{Tf}\right) \\ g^{Tf} - x_0^{Tf} = k_1\left(g^t - x_0^{Tf}\right) + k_2\left(g^o - x_0^{Tf}\right) \end{cases} . \tag{39}$$

According to the geometric relationship in Fig.3 , we can get $g^{Tf} - x_0^{Tf} = \Delta x^o - \Delta x^t$. Taking the conditions into (39), we have

$$\left(I - k_1\right)\left(I + k_2\right)^{-1} = tr\left(\Delta x^t \left(\Delta x^o\right)^H\right), \tag{40}$$

which means k_1 and k_2 are determined by the relative distance of the start and end contact points. Using (23), we have

$$x^{Tf} - k_1 x^t = k_2 x^o + \left(I - k_1 - k_2\right)x_0^{Tf} . \tag{41}$$

Furthermore, we assume that x^t and x^o can be expressed by DMP, sharing the same phase variable s as

$$\begin{cases} \tau\dot{v}^t = \alpha_z\left(\beta_z\left(g^t - x^t\right) - v^t\right) + f^t\left(s\right) \\ \tau\dot{x}^t = v^t \end{cases} , \tag{42}$$

$$\begin{cases} \tau\dot{v}^o = \alpha_z\left(\beta_z\left(g^o - x^o\right) - v^o\right) + f^o\left(s\right) \\ \tau\dot{x}^o = v^o \end{cases} . \tag{43}$$

Using (22) and (42), we have

$$\begin{cases} \tau\dot{\tilde{v}}^t = \alpha_z\left(\beta_z\left(\tilde{g}^t - \tilde{x}^t\right) - \tilde{v}^t\right) + f^{Tf}\left(s\right) - k_1 f^t\left(s\right) \\ \tau\dot{\tilde{x}}^t = \tilde{v}^t \end{cases} , \tag{44}$$

where $\tilde{g}^t = g^{Tf} - k_1 g^t$, $\tilde{x}^t = x^{Tf} - k_1 x^t$ and $\tilde{v}^t = v^{Tf} - k_1 v^t$.

Taking (41) into (44), we can get

$$k_2\tau\dot{v}^o = \alpha_z\left(\beta_z\left(\tilde{g}^t - k_2 x^o\right) - k_2 v^o\right) + f^{Tf}\left(s\right) - k_1 f^t\left(s\right) + \alpha_z\beta_z\left(I - k_1 - k_2\right)x_0^{Tf} . \tag{45}$$

Comparing (43) with (45) and considering $g^{Tf} - x_0^{Tf} = \Delta x^o - \Delta x^t$ we have

$$f^{Tf}\left(s\right) - k_1 f^t\left(s\right) - k_2 f^o\left(s\right) = 0 . \tag{46}$$

Taking $f^t\left(s\right) = f^{Tf}\left(s\right)/k_1 - k_2 f^o\left(s\right)/k_1$ into (42), (25) is achieved.

C. Proof of Theorem 1

Following (31), we set $V_1^c = \frac{1}{2}\log\frac{\left(\overline{h}_i - \underline{h}_i\right)^2}{4\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)}$ and $V_2^c = \frac{1}{2}k_n\tau z_2^2$. As $\left(\overline{h}_i - \underline{h}_i\right)^2 \geq 4\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right) > 0$, then

$$\begin{cases} V_1^c \geq \dfrac{1}{2}\log\left(1\right) = 0 \\ V_2^c \geq 0 \end{cases} , \tag{47}$$

Thus, $V^c = V_1^c + V_2^c > 0$ and the desired condition is $\dot{V}^c < 0$.

Taking (29) into $\dot{V}_1^c$, we have

$$\begin{aligned} \dot{V}_1^c &= -\frac{z_1 - \left(\overline{h}_i + \underline{h}_i\right)/2}{\tau\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)}\left(\tau\dot{x}^u - \tau\dot{x}^N\right) \\ &= -\frac{z_1 - \left(\overline{h}_i + \underline{h}_i\right)/2}{\tau\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)}\left(\tau\dot{x}^u - z_2 - \lambda\right) \\ &= -\frac{z_1 - \left(\overline{h}_i + \underline{h}_i\right)/2}{\tau\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)}\left(\tau\dot{x}^u - \lambda - z_1 + \frac{\overline{h}_i + \underline{h}_i}{2}\right) - \\ &\quad \left(z_1 - \frac{\overline{h}_i + \underline{h}_i}{2} - \frac{z_2}{2}\right)^2 \Big/ \tau\left(\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)\right) + \\ &\quad \frac{z_2^2}{4\tau\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)} \end{aligned} . \tag{48}$$

Setting $\tau\dot{x}^u - \lambda - z_1 + \left(\overline{h}_i + \underline{h}_i\right)/2 = 0$, we can get

$$\lambda = \tau\dot{x}^u - z_1 + \left(\overline{h}_i + \underline{h}_i\right)/2 , \tag{49}$$

and the following inequality is achieved by (48) as

$$\dot{V}_1^c \leq \frac{z_2^2}{4\tau\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)} . \tag{50}$$

The time derivative of V_2^c is

$$\dot{V}_2^c = k_n\tau z_2\dot{z}_2 = k_n\tau z_2\left(\dot{v}^N - \dot{\lambda}\right). \tag{51}$$

The sufficient condition for $\dot{V}^c \leq 0$ is $\forall\, k_1 > 0$ satisfies

$$\begin{aligned} \dot{V}_1^c + \dot{V}_2^c &\leq k_n\tau z_2\left(\dot{v}^N - \dot{\lambda}\right) + \frac{z_2^2}{4\tau\left(\overline{h}_i - z_1\right)\left(z_1 - \underline{h}_i\right)} \\ &\leq 0 \end{aligned} . \tag{52}$$

Taking the definition of λ into (52), we have

$$k_n \tau z_2 \left(\dot{v}^N - \dot{v}^u + \dot{z}_1 \right) \le -\frac{z_2^2}{4\tau \left(\overline{h}_i - z_1 \right)\left(z_1 - \underline{h}_i \right)} = -\left(k_z + \frac{1}{\tau \left(\overline{h}_i - \underline{h}_i \right)^2} \right) z_2^2 . \quad (53)$$

where k_z is a tiny positive number.

Finally, taking (21) and (29) into (53), we can get

$$\Delta u^c = \alpha_z \beta_z z_1 + \left(\alpha_z \tau - \tau \right) \dot{z}_1 - \left(k_z + \frac{1}{\tau \left(\overline{h}_i - \underline{h}_i \right)^2} \right) \frac{z_2}{k_n} . \quad (54)$$

REFERENCES

[1] A. Billard, and D. Kragic, "Trends and challenges in robot manipulation, " *Science*, vol. 364, no. 6446, p.eaat8414. 2019.

[2] M. Qin, J. Brawer, and B. Scassellati, "Robot tool use: A survey". *Front. Robot. AI,* vol. 9, p.1009488. 2023.

[3] R. A. Knepper, T. Layton, J. Romanishin and D. Rus, "IkeaBot: An autonomous multi-robot coordinated furniture assembly system," in *IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 855-862.

[4] K. Pfeiffer, A. Escande, and A. Kheddar, "Nut fastening with a humanoid robot, " in *IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Sep. 2017, pp. 6142-6148,

[5] R. Li, D.T. Pham, J. Huang, Y. Tan, M. Qu, Y. Wang, M. Kerin, K. Jiang, S. Su, C. Ji, and Q. Liu, "Unfastening of hexagonal headed screws by a collaborative robot, " *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp.1455-1468, 2020.

[6] X. Mu, Y. Xue and Y. -B. Jia, "Robotic Cutting: Mechanics and Control of Knife Motion," in *Int. Conf. Robot. Autom.*, Montreal, QC, Canada, 2019, pp. 3066-3072.

[7] Y. Xue and Y. -B. Jia, "Gripping a Kitchen Knife on the Cutting Board," in *IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Las Vegas, NV, USA, 2020, pp. 9226-9231.

[8] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 3, pp. 297-330, 2020.

[9] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration". *Rob. Auton. Syst.*, vol. 57, no. 5, pp. 469-483, 2009.

[10] A. G. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Rob. Auton. Syst.*, vol. 54, no. 5, pp.370-384. 2006.

[11] M. Deniša, Gams, A., Ude, A. and Petrič, T., "Learning compliant movement primitives through demonstration and statistical generalization," *IEEE/ASME Trans. Mech.*, vol.21, no. 5, pp.2581-2594, 2015.

[12] H. Qiao, Y. X. Wu, S.L. Zhong, P. J. Yin, and J. H. Chen, "Brain-inspired intelligent robotics: Theoretical analysis and systematic application, " *Machine Intelligence Research*, vol. 20, no. 1, pp.1-18, 2023.

[13] A. Xie, F. Ebert, S. Levine, and C. Finn, "Improvisation through physical understanding: Using novel objects as tools with visual foresight," *arXiv preprint arXiv:1904.05538*, 2019.

[14] S. Elliott, Z. Xu and M. Cakmak, "Learning generalizable surface cleaning actions from demonstration," *in IEEE Int. Symp. Robot Human Int. Comm. (RO-MAN)*, 2017, pp. 993-999.

[15] L. H. Kong, W. He, W. S. Chen, H. Zhang, and Y. N. Wang, "Dynamic movement primitives based robot skills learning, " *Machine Intelligence Research*, vol. 20, no. 3, 396-407, 2023.

[16] C. Yang, C. Zeng, C. Fang, W. He, and Z. Li, "A DMPs-based framework for robot learning and generalization of humanlike variable impedance skills," *IEEE/ASME Trans. Mech.*, vol. 23, no. 3, pp. 1193-1203, 2018.

[17] K. Yamazaki et al., "System integration of a daily assistive robot and its application to tidying and cleaning rooms," in *IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Taiwan, 2010, pp. 1365-1371,

[18] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear Dynamic systems." in *Proc. IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Maui, Hawaii, 2001, pp. 752-757.

[19] F. Bian, D. Ren, R. Li, P. Liang, K. Wang, and L. Zhao, "An extended DMP framework for robot learning and improving variable stiffness manipulation, " *Ass. Autom.* vol. 40, no. 1, 2020, pp. 85-94.

[20] Z. Lu, and N. Wang, "Dynamic movement primitives based cloud robotic skill learning for point and non-point obstacle avoidance," *Assem. Autom.*, vol. 41, no. 3, 2021, pp. 302-311.

[21] Z. Hong, S. Bian, P. Xiong and Z. Li, "Vision-Locomotion Coordination Control for a Powered Lower-Limb Prosthesis Using Fuzzy-Based Dynamic Movement Primitives," *IEEE Trans. Autom. Sci. Eng.*, 2023.

[22] R. Huang, H. Cheng, J. Qiu and J. Zhang, "Learning Physical Human–Robot Interaction With Coupled Cooperative Primitives for a Lower Exoskeleton," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1566-1574, Oct. 2019.

[23] X. Hao, Z. Li, P. Huang, P. Shi and G. Li, "Hierarchical Task-Oriented Whole-Body Locomotion of a Walking Exoskeleton Using Adaptive Dynamic Motion Primitive for Cart Pushing," *IEEE Trans. Autom. Sci. Eng.*, 2023.

[24] C. Zou, R. Huang, J. Qiu, Q. Chen and H. Cheng, "Slope Gradient Adaptive Gait Planning for Walking Assistance Lower Limb Exoskeletons," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 405-413, April 2021.

[25] H. Su, A. Mariani, S. E. Ovur, A. Menciassi, G. Ferrigno and E. De Momi, "Toward Teaching by Demonstration for Robot-Assisted Minimally Invasive Surgery," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 484-494, April 2021.

[26] R. Prakash and L. Behera, "Neural Optimal Control for Constrained Visual Servoing via Learning From Demonstration," *IEEE Trans. Autom. Sci. Eng.*, 2023.

[27] Z. Lu, N. Wang and C. Yang, "A constrained DMPs framework for robot skills learning and generalization from human demonstrations," *IEEE /ASME Trans. Mech.*, vol. 26, no. 6, pp. 3265-3275, Dec. 2021.

[28] C. Zeng, C. Yang, H. Cheng, Y. Li, and S.-L. Dai, "Simultaneously encoding movement and sEMG-based stiffness for robotic skill learning," *IEEE Trans. Industr. Inform.*, vol. 17, no. 2, pp. 1244-1252, 2020.

[29] C. Yang, C. Zeng, Y. Cong, N. Wang, and M. Wang, "A learning framework of adaptive manipulative skills from human to robot," *IEEE Trans. Industr. Inform.* vol. 15, no. 2, pp. 1153-1161, 2018.

[30] D. H. Zhai, Z. Xia, H. Wu and Y. Xia, "A motion planning method for robots based on DMPs and modified obstacle-avoiding algorithm," *IEEE Trans. Autom. Sci. Eng.,* vol. 20, no. 4, pp. 2678-2688, Oct. 2023.

[31] H. Kim, C. Oh, I. Jang, S. Park, H. Seo and H. J. Kim, "Learning and generalizing cooperative manipulation skills using parametric dynamic movement primitives," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3968-3979, Oct. 2022.

[32] D. H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots*, 2008, pp. 91–98.

[33] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired Dynamic systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance." in *IEEE Int. Conf. Robot. Autom.*, Kobe, Japan, May 12, 2009, pp. 2587-2592.

[34] È. Pairet, P. Ardón, M. Mistry, and Y. Petillot, "Learning generalizable coupling terms for obstacle avoidance via low-dimensional geometric descriptors," *IEEE Robot. Autom. Lett.,* vol. 4, no. 4, pp. 3979–3986, Oct. 2019

[35] J. Umlauft, D. Sieber, and S. Hirche "Dynamic movement primitives for cooperative manipulation and synchronized motions," in *Proc. IEEE Int. Conf. Robot. Autom*., 2014, pp. 766–771

[36] T. Kulvicius, M. Biehl, M. J. Aein, M. Tamosiunaite, and F. Wörgötter, "Interaction learning for dynamic movement primitives used in cooperative robotic tasks," *Robot. Auton. Syst.,* vol. 61, no. 12, pp. 1450–1459, 2013.

[37] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 816–830, Aug. 2014.

[38] H. Tan, E. Erdemir, K. Kawamura, and Q. Du, "A potential field method-based extension of the dynamic movement primitive algorithm for imitation learning with obstacle avoidance." in *Inter. Conf. Mech. Autom.*, Beijing, China, 2011, pp. 525-530.

[39] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives," *Robot. Auto. Syst.*, 2011. Vol. 59, no. 11, pp.910-922.

[40] Hassanin, M., Khan, S. and M. Tahtali, "Visual Affordance and Function Understanding: A Survey, " *ACM Computing Surveys (CSUR),* vol.54, no. 3, pp.1-35. 2021.
[41] Y. Zhu, Y. Zhao, and S. C. Zhu, "Understanding tools: Task-oriented object modeling, learning and recognition." in *Proc. IEEE Conf. Comp. Vision Patt. Recogn.*, Boston, MA, USA, 2015, pp. 2855-2864.
[42] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese, "Keto: Learning keypoint representations for tool manipulation, " in *IEEE Int. Conf. Robot. Autom,* Paris, France, May. 2020, pp. 7278-7285.
[43] S. Akizuki, and Y. Aoki, "Tactile Logging for Understanding Plausible Tool Use Based on Human Demonstration, " in *BMVC*, September 2018, pp. 334-345.
[44] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance Detection of Tool Parts from Geometric Features, " in *IEEE Int. Conf. Robot. Autom*, Washington WA, USA, May 2015, pp.1374–1381.
[45] Saito, N., Ogata, T., Mori, H., Murata, S. and Sugano, S., "Tool-use model to reproduce the goal situations considering relationship among tools, objects, actions and effects using multimodal deep neural networks," *Front. Robot. AI*, vol.8, 2021. pp.309.
[46] J. Yuan, C.-M. Chew, and V. Subramaniam, "Learning Geometric Constraints of Actions from Demonstrations for Manipulation Task Planning." in *IEEE Int. Conf. Robot. Biom.*, Kuala Lumpur, Malaysia, 2018, pp. 636-641.
[47] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations." In *Int.l Workshop Algorithm. Found. Robot.,* Mérida, México, Dec 9 2018, pp. 228-245.
[48] G. Chou, N. Ozay, and D. Berenson, "Learning parametric constraints in high dimensions from demonstrations." in *Conf. Rob. Learn.*, London, UK, May 2020, pp. 1211-1230.
[49] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *Int. J. Rob. Res.*, vol. 38, no. 7, pp. 833-852, 2019.
[50] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision," *Int. J. Rob. Res.*, vol. 39, no. 2-3, pp. 202-216, 2020.
[51] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese, "Keto: Learning keypoint representations for tool manipulation," in *IEEE Int. Conf. Robot. Autom.*, 2020, May, pp. 7278-7285.
[52] K. P. Tee, J. Li, L. T. Pang Chen, K. W. Wan and G. Ganesh, "Towards Emergence of Tool Use in Robots: Automatic Tool Recognition and Use Without Prior Tool Learning," in *IEEE Int. Conf. Robot. Autom.*, Brisbane, Australia, 2018, pp. 6439-6446.
[53] S. Brown, and C. Sammut, "A relational approach to tool-use learning in robots." in *Int. Conf. Ind. Logic Progr.*, Dubrovnik, Croatia, Sep 17, 2012, pp. 1-15.
[54] X. Li, Z. Sun, D. Cao, Z. He and Q. Zhu, "Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms, and Verifications," *IEEE/ASME Trans. Mech.* vol. 21, no. 2, pp. 740-753, April 2016.
[55] P. Boonvisut and M. C. Çavuşoğlu, "Estimation of Soft Tissue Mechanical Parameters From Robotic Manipulation Data," *IEEE/ASME Trans. Mech.* vol. 18, no. 5, pp. 1602-1611, Oct. 2013
[56] K. P. Tee, S. Cheong, J. Li, and G. Ganesh. "A framework for tool cognition in robots without prior tool learning or observation. *" Nature Mach. Intell.*, 2022, pp.1-11.
[57] S. M., Khansari-Zadeh, and A. Billard, "A Dynamic system approach to realtime obstacle avoidance. " *Auton Robot*, vol. 32, pp. 433–454, 2012.
[58] N. Wang, C. Chen, and Di A., Nuovo, "A framework of hybrid force/ motion skills learning for robots. " *IEEE Trans. Cogn. Dev. Syst.*, vol. 13, no. 1, pp.162-170. 2020.
[59] Z. Lu, N. Wang, M. Li and C. Yang, "Incremental Motor Skill Learning and Generalization From Human Dynamic Reactions Based on Dynamic Movement Primitives and Fuzzy Logic System," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 6, pp. 1506-1515, June 2022.
[60] M. Ginesi, N. Sansonetto, and P. Fiorini, "Overcoming some drawbacks of dynamic movement primitives," *Rob. Auton. Syst.*, vol. 144, pp. 103844, 2021.
[61] S. Karaman, and E. Frazzoli, "Sampling-based algorithms for optimal motion planning, " *Int. J. Rob. Res.*, vol. 30, no. 7, pp.846-894. 2011.
[62] K. Takahashi, K. Kim, T. Ogata, and S. Sugano, "Tool-body assimilation model considering grasping motion through deep learning, " *Rob. Auton. Syst.*, 91, pp.115-127, 2017.
[63] D. Seita, Y. Wang, S.J. Shetty, E.Y. Li, Z. Erickson, and D. Held, " Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds, " in *Conf. Rob. Learn.* , pp. 1038-1049, PMLR. March, 2023.
[64] A. Byravan and D. Fox, "SE3-nets: Learning rigid body motion using deep neural networks," *IEEE Int. Conf. Robot. Autom*, Singapore, 2017, pp. 173-180
[65] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan, "Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools, " *arXiv preprint arXiv:2203.17275*. 2022
[66] M. Qin, "Robot tool use: learning, transferring, reasoning, and applying knowledge about robots using human tools" Doctoral dissertation, Yale University, 2022.
[67] M. Eppe, P. D. Nguyen, and S. Wermter, "From semantics to execution: Integrating action planning with reinforcement learning for robotic causal problem-solving, " in *Front. Robot. AI*, 6, p.123. 2019.
[68] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *IEEE Int. Conf. Robot. Autom*, pp.3828-3834, May, 2011.
[69] Z. Jin, W. Si, A. Liu, W. -A. Zhang, L. Yu and C. Yang, "Learning a Flexible Neural Energy Function With a Unique Minimum for Globally Stable and Accurate Demonstration Learning," in *IEEE Trans. Robot.*.
[70] C. Zito, R. Stolkin, M. Kopicki, and J. L. Wyatt, "Two-level RRT planning for robotic push manipulation." in *Proc. IEEE/RSJ Int. Conf. Int. Robot. Syst.*., 2012, October, pp. 678-685.
[71] H. Lee, H. Kim and H. J. Kim, "Planning and Control for Collision-Free Cooperative Aerial Transportation," in *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 189-201, Jan. 2018.
[72] L. Y. Chen et al., "AutoBag: Learning to Open Plastic Bags and Insert Objects," *IEEE Int. Conf. Robot. Autom*, London, UK, 2023, pp. 3918-3925,
[73] K.,Pertsch, Y. Lee, and J., Lim, "Accelerating reinforcement learning with learned skill priors, " in *Conf. rob. Learn.*, pp. 188-204, PMLR. October. 2021,
[74] G. Wang, M. Xin, W. Wu, Z. Liu and H. Wang, "Learning of Long-Horizon Sparse-Reward Robotic Manipulator Tasks With Base Controllers," in *IEEE Trans. Neural Netw. Learn. Syst.*, 2022

Zhenyu Lu (M'21) received the Ph.D degree in Northwestern Polytechnical University, Xi'an, China in 2019. He is currently working as a senior research fellow in Bristol Robotic Laboratory & University of the West of England, Bristol. His research interests include teleoperation, human–robot interaction and learning.

Ning Wang (S'07-M'11) received the M.S. and Ph.D. degrees in electronics engineering from the Chinese University of Hong Kong, Hong Kong, in 2007 and 2011, She is a Senior Lecturer in Robotics at the Bristol Robotics Laboratory, University of the West of England, United Kingdom. Her research interests lie in signal processing, intelligent data analysis, human-robot interaction and autonomous driving.

Chenguang Yang (Fellow, IEEE) received the B.Eng. degree in measurement and control from Northwestern Polytechnical University, Xian, China, in 2005, and the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010. He performed postdoctoral studies in human robotics at the Imperial College London, London, U.K from 2009 to 2010. He is Chair in Robotics with Department of Computer Science, University of Liverpool, UK. He was

awarded UK EPSRC UKRI Innovation Fellowship and individual EU Marie Curie International Incoming Fellowship. As the lead author, he won the IEEE Transactions on Robotics Best Paper Award (2012) and IEEE Transactions on Neural Networks and Learning Systems Outstanding Paper Award (2022). He is the Corresponding Co-Chair of IEEE Technical Committee on Collaborative Automation for Flexible Manufacturing. His research interest lies in human robot interaction and intelligent system design.