# A Novel Dynamic Movement Primitives-based Skill Learning and Transfer Framework for Multi-Tool Use *

Zhenyu Lu, Ning Wang, Miao Li, Chenguang Yang, *Senior Member, IEEE*

***Abstract*— Dynamic Movement Primitives (DMPs) is a general method for learning skills from demonstrations. Most previous research on DMP has focused on point to point skill learning and training, and the skills learned are usually generalized based on the same tool or manipulator. There is rare research on skill learning and transfer between two or more different tools. For this problem, a new DMP-based skill learning and transfer framework is proposed for the use of multiple tools. It consists of two types of skills: Object Effective (OE) skills and State Switching (SS) skills. OE skills consider the tools' limited forcing areas that can be expressed as constrained inequalities, and extract skills from demonstrations. It can then be generalized along with changes in the shape and range of influence of a new tool. SS skill is used to connect OE skills and implement changes of contact points of the object and tool. Finally, the two skills are integrated and used to realize the transfer of skills from the demonstrated tool to the new tool. An experiment is conducted to verify the effectiveness of the proposed framework, and the procedural solutions and the final manipulation effect are shown in detail.**

## I. INTRODUCTION

Tools such as hammers, screwdrivers, and wrenches etc. are invented for various purposes and are used to help people accomplish various tasks. Some tools are designed with special functions and others are multi-functional. Enabling the robots to use tools to flexibly complete manipulation tasks as humans is always a challenge for current research. Tools can extend human reaches and amplify the physical strength, which have the same meaning for robots. But how to enable robots to know and use tools by itself, some previous research provides technical ideas.

Brown and Sammut et. al concluded four key aspects for learning the task-oriented tool usages: (a) understanding the desired effect; (b) identifying properties of an object as a tool (c) determining the correct orientation of the tool prior to usage; and (d) manipulating the tool [1]. Some previous research based on artificial intelligence and computer vision have proposed solutions for the former three aspects, e.g. Xie et.al [2] discussed prior approaches of tool-use and proposed robotics learning methods and concluded three categories for analyzing the tools' uses: analytic model-based method, direct learning, and learning from demonstration (LfD). For tool's understanding, some recent work has studied using geometric analytic model or cognitive learning methods to recognize the effective forcing areas and basic properties of tools, and used the recognized results for tools' selection [12], performing task planning [15] and policy learning [11]. The main problem of analytic model's is the weak scalability and accumulating errors that are not considered in the modelling. Compared with direct learning, LfD provides a more natural and easy way such that it draws a lot of attention in recent years [3], which also can provide a serious of actions for tool's manipulations. Based on our previous research [9],[10], [14], this paper will focus on robot skill learning from demonstrations and transfer for multi-tool use manipulations. Although there is some previous research about tool use using LfD technology [16]-[18] , to our best knowledge, there is little research has been conducted on transferring skills between different tools.

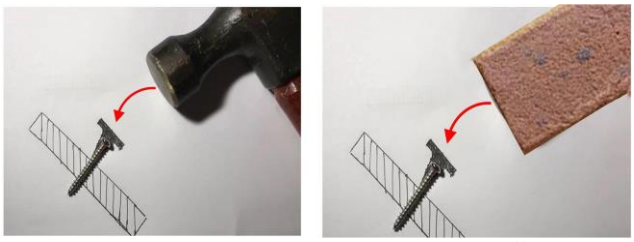


Figure 1. Mulit-tool use to achieve the same manipulation effect (the right figure is modified from the left for illustration)

As mentioned by Brown et al. [1] and Fang et al. [10], the operational tools' usages can be recognized and the tools are classified into T-shape and L-shape etc., but the learned skills in one tool are hard to be applied directly on another. However, as humans, we use a hammer to hit a nail (Fig. 1). If there is no hammer around, we can use a brick instead to achieve the same effect (Fig. 1), which means that we acquire the '**pure skill' of hitting**, but not the skill of using hammer to hit. Thus, the basic idea of this paper is to learn a 'pure skill' from the demonstrations of tool's interactions and applied the learned results to a new tool-use case. The proposed framework is based on the Dynamic Movement Primitives (DMPs), a very general method for robotic skill learning. It was firstly proposed by Ijspeert et al [4] to extract a trajectory in a similar

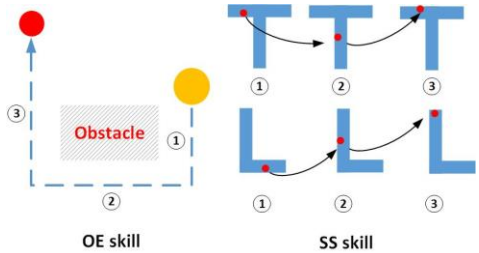



Figure 2. Tools used by humans following a certain trajectory (skills)

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

Zhenyu Lu, Ning Wang and Chenguang Yang are with the Bristol Robotics Laboratory, University of the West of England, Bristol, UK (corresponding author to provide cyang@ieee.org ). Miao Li is with HONG YI HONOR COLLEGE of Wuhan University, Wuhan, China (e-mail: miao.li@whu.edu.cn)

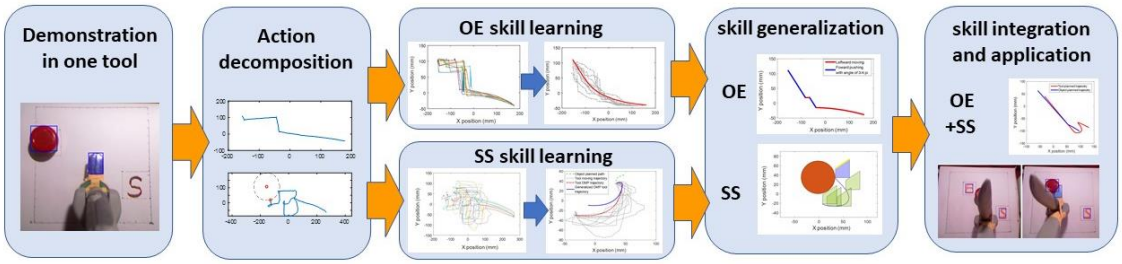


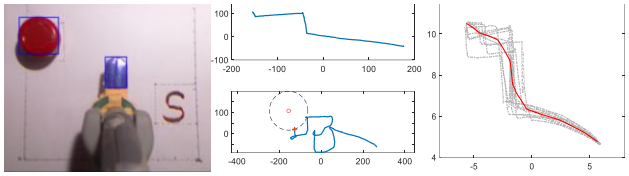

Figure 3. Diagram of Dynamic Movement Primitives (DMP)-based skill Learning and transfer framework for multi-tool use

form to the demonstrations without the need for kinematic modelling. DMP was improved and widely utilized for the tasks such as object movement tracking [5], handwriting [6], [22], and clothes manipulation [7] etc. There is some work using DMP for tool-use [8], [9] case, but as mentioned before, the skills are learned and tested using the same manipulator or tools.

The proposed framework consists of two types of skills: The Object Effective skill (OE) and the State Switching skill (SS). The function of the OE skill is to move the object along the intended trajectory under the influence of tools. Learning the OE skill is to acquire a pure skill that can be changed along with the tools that have different shapes and effective force ranges. In Fig.2, the orange object is moved over the obstacle with line shadow to reach the target point (big red point circle). The motion path consists of three segments, labeled 1 to 3. The right figure in Fig.2 shows the possible contact points (red dots) on two different tools (T-shape and L-shape) in each step to realize the manipulation effect in the left figure. The inversion of the contact points is completed by the SS skill (black arrow lines). SS skill mainly considers the influence of the outline of the tool and the object to reduce the possibilities of conflicts. Therefore, a learned SS skill can be generalized to adapt to tools with new shapes or change the contact regions in different tools.

Fig.3 shows the diagram of the DMP-based skill Learning and transfer framework for multi-tool use. First, by using the demonstration, we decompose the actions and create two date-sets for the OE and SS skills, the two skills are learning using different methods. For a new task in a new environment with a different tool to the demonstrated one, we generalize the skill according to the starts, destinations and scalar factors and re-shape the trajectories considering new situations e.g. obstacles and tool's special shape. Finally, both OE and SS skills are integrated to complete the interactions and adaptations of the tool to a new task.

The reminder of this paper is organized as follows: Section II provides a brief introduction to the knowledge of DMP and illustrates the main challenges for learning and transferring skills in handling multiple tools by using the original DMP through a preliminary experiment. In section III, two subsections independently introduce the skills OE and SS. In section IV, we conduct an experiment on a small platform consisting of a Phantom Omni joystick, a Kinect, and two homemade tools and an object to test the effectiveness of the proposed methods. The conclusions from the phases and the final results are presented in detail. The last section contains a final conclusion.



(a) (b) (c)

Figure 4. Experimeal setup and interface and the recorded trajectories of the object and the tool (with Omni joystick) **(a)** physical platform with a red object and a blue finger-shape tool **(b)** trajctory of the object center (upper figure) and the manipulator's end (lower figure) **(c)** trajecotry learned results by using the original DMP that the corners between the line segements are erased due to the minimizing function (4).

## II. PRELIMINARY WORK

### A. *Original formulation of DMPs*

The DMP model [4] is presented as

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

where $x$ and $v$ represent the position and the velocity in one dimension, and $f\left( s \right) = \theta^T \Psi(s)$ is the linear combination of nonlinear radial basis functions with $\Psi(s) = \left[ \psi_1, \psi_2, ..., \psi_n \right]^T$, and $\theta = \left[ w_1, w_2, ..., w_n \right]^T$ are the weights, and

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

where $c_i$ and $h_i > 0$ are the centers and widths of radial basis functions $\varphi_i(s)$, and $\alpha_z, \beta_z > 0$ are coefficients of the linear part in (1) and (1) has the unique attractor point at $x = g$, $v = 0$. $\tau > 0$ is a timing parameter for adjusting speed before execution of movements, and $s$ is a phase variable to achieve dependency of function $f\left( s \right)$ out of time. The dynamics of $s$ is expressed by a canonical system as

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

The convergence time of $s$ is determined by the factor $\gamma$. If $s \to 0$, the value of $f\left( s \right)$ trends to 0 and $\left[ x, v \right]$ will reach to the final point $[g, 0]$. Vector $\theta$ is learned using supervised learning algorithms by minimizing the error function:

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

where $f(s)$ represents the final calculated forcing function, and $f_j^{Tar}(s)$ is the target value calculated by the *jth* trajectory

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

where $x_j$ and $v_j$ represent the positions and velocities of the *jth* trajectory.

### B. Problem description and assumptions

Fig. 3 (a) shows the experimental conditions from the top view. Human operator handles the pen of the Joystick attached with a tool (blue part) to contact and move the object (red part) to reach to the target square. Due to the **operator's preference and the tool's shape**, only the upper and left sides of the tool can be used for manipulation (as shown in Fig.4(a)). Thus, the object can only be **pushed leftward and forward**. The upper subfigure of Fig.3 (b) shows a trajectory of object center that is consisted of several line segments. If the object is grasped by human hands or attached directly on the robot's end, it can be moved straight from the start to the target and the angles between the line segments can be avoided. Thus, the first problem can be described as:

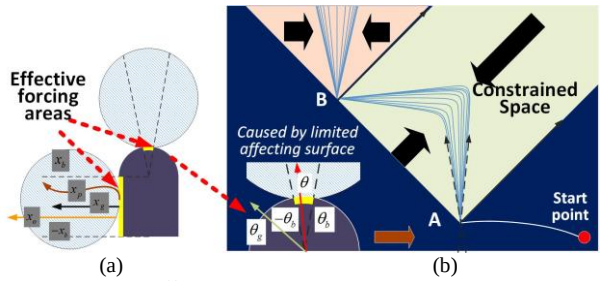
**Problem 1:** If we use the demonstrations under the influence of user's preferences and tool's usages for skill training, it is easy to bring these characteristics to the skill generalization process that may be not suitable to the new tool or new cases. Additionally, if we use the original DMP formulation in (1) for skill learning, because there is no any limitation on the learned trajectories, these properties will be erased by the minimizing calculation (Fig. 3(c)) in (4).

As tool-use is a process interacted by the tool and the object /environment, we also record the tool's moving trajectory (the end of the manipulator, and the tool is fixed with constant relationship to the manipulator). For changing contact regions (from tools' top to the left or vice versa), there are two arching trajectories. So the second problem is

**Problem 2:** how to learn the skill for switching contact points. During the switching process, avoiding conflicts between the tool and object is probably the main concern.

To address above problems, we propose the assumptions:

**Assumption 1**: The potential contact regions and properties of tools can be recognized by using the methods in [12], or some parameter identification methods e.g. in [13]. In this work, it is assumed that these properties and regions are known or manually determined before skill learning and generalization.




(a) (b)

Figure 5. limited effective tool areas and potential trajectories under the influence of the limitations **(a)** Effective force areas of the tool at the upper and left side of the tool (yellow coloured lines) **(b)** Potential trajectories are generated within the limited space under the influence of the tool limitations. The figure on the lower left shows the constraints of the upper contact area. The figure above right shows the effective trajectories (blue lines) from the starting point to the intersection point A and point B.

**Assumption 2:** All the tools in this paper are basic tools that can be summarized as limited shapes introduced in [10], such as I shaper, T-shape and L-shape etc.

## III. DMP-BASED SKILL LEARNING FRAMEWORK

For above two problems, we propose the OE skill and SS skill separately, both of which are based on DMP and used for multi-tool use scenes.

### A. Object effective (OE) skill learning

As mentioned above, the effective attack surfaces are on the top and left side of the tool, selected by the operator to produce forward and left thrusts. While for an object attached to the end of the robot or gripped by the robot hand strictly, it can be moved with the robot. The limited effective ranges of the tool constrain the directions of the object's motion. Let us take Fig. 5(b) as an example: the hands can be used to move the object from the starting point to points A and B, which happen to be within the dark blue and light blue regions called "universal space". If we set point A and point B as internal passing points to reach to target, we can choose two ways to move the object from A to B: first leftward and then forward up to B or the forward first. Here, we choose the latter strategy and the light blue universal space is compressed into a space with blue lines such that there are several corners along the final trajectory, as it is shown in Fig. 4(b).

The pure OE skill is expected to remove these constraints and to be learned in the universal space. Then, for a new task using a new-shape tool, the pure OE skill will be generalized first and condensed again according to the constrains generated by the new tool. Therefore, before using OE skills learning method, we assume that the effective force regions and tool's shapes are known, and the constrains generated by these conditions can be expressed by several inequalities.

Then, what we do is to build a nonlinear mapping function for the trajectory points in the universal space and condensed space. We classify these constraints into angle constraints and Cartesian space constraints. The typical angle constraint is the one affected on the top of the tool in Fig.5 (a) and the effective pushing directions are within a small range colored in yellow and shown in Fig. 5(b). Here, we refer our previous research [10] and set an angle condensed in the constrained/condensed space as $\theta$ and $\theta_b$ as the boundary condition of $\theta$, and the angle released in the universal space is $\theta_g$, then a nonlinear mapping function is built as

$$\theta = sign(\theta_g) \bullet \left( \theta_b - e^{-k\frac{|\theta_g|}{\theta_b}} \theta_b \right), \qquad (6)$$

where $k \in (0,1)$ is an adjusting factor. According to the proofs in [19], $\theta \in (-\theta_b, \theta_b)$ is strictly satisfied.

For the transformation in the Cartesian space, we set $x_o$ as a referring point, which are within constraints, as presented in

our previous work of [10]. Set $x_p$ as a point in the universal space, $x_b$ as a point on the constrained boundary, and $x_g$ as a point in the compressed space. As it is shown in Fig.5 (a), the leftward push action is constrained by the effective pushing points aligned on the tool's left side. Define $r_p = |x_p - x_o|$, $r_b = |x_b - x_o|$, $r_g = |x_g - x_o|$, and $k \in (0,1)$, a transformation function (7) is built to generate a constrained trajectory within an area or out of an area as

$$\begin{cases} r_g = r_b + e^{-k\frac{r_p}{r_b}} r_b & outerspace \\ r_g = r_b - e^{-k\frac{r_p}{r_b}} r_b & innerspace \end{cases}. \quad (7)$$

Eqs. (6) and (7) are used to condense a trajectory in the universal space into a constrained space. We can use inverse calculation based on Taylor expansion (usually use two order expansion) to expand $\theta$ or $x_g$ from the condensed space to the universal space and achieve $\theta_g$ and $x_p$. For example, using the Taylor expansion of (6), a constrained angle $\theta$ is transformed into angle $\theta_g$ in the universal space by

$$|\theta_g| = k \pm \frac{\sqrt{0.25\theta_b^2 - \theta\theta_b}}{k}, \quad (8)$$

which is determined by $k$ and has four possible solutions for the same $\theta$. We use evaluation principles such as smoothness of the generated trajectory to select one as the final universal point. The OE skill learning and generalization procedure can be summarized as following steps:

1. Select constraint type and build the mapping function (6) or (7);
2. Use demonstrated trajectory and the inverse calculation of (6) and (7), like (8), to calculate the results in the universal space;
3. Use (1) to learn from the transformed trajectory points and achieve skills in the universal space;
4. Use mapping function (6) or (7) again and constraints of new tools and new environmental conditions to condense the learned skill into the constrained space.

**Remark 1**: The relative distance between two adjacent points is increased and compressed by the nonlinear transformation functions, which may lead to nonuniformity of the generalized trajectory. We can set up an optional point tank containing multiple points near the center points for point selection to ensure the smoothness of the trajectory.

*B. State switching (SS) skills learning*

Fig. 6 shows an illustration of the contact state switching process. A tool is moved from the right contact surface on the object to the lower contact surface. Not only the object but also the contact areas of the tool are switched from left to top. Therefore, the change trajectory (black dashed line in Fig. 6) has to do with both the object and the multifunction tool. Even for the same object, the contact points will be different for different tools due to the different shapes, and the ability of the tool to change state will be different. We would like to split the one-point trajectory of the tool (black dashed line) into a virtual object motion trajectory (red line) and a pure tool motion capability (red dashed line), where the latter can be generalized depending on the changes in the shape and contact areas of the tool. Similar to OE, we assume that the contact points of the tool and object as well as the outlines of the tool and object are precisely known.

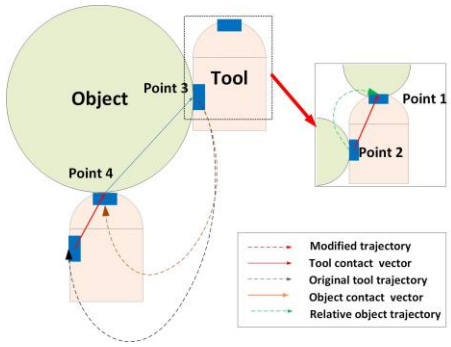



Figure. 6. Illustration of tool state switching process

Assuming during the tool's state switching process, object keeps static. Set the initial contact points (point 3 on the object and point 2 on the tool) locate at $\mathbf{x}_o$ and the vector between point 2 and point 3 on the object as $\Delta\mathbf{x}^o$, the vector from point 2 to point 1 as $\Delta\mathbf{x}^t$, the final position of point 2 as $\mathbf{g}$, the final position of point 4 as $\mathbf{g}^O$ and the initial position of point 1 as $\mathbf{g}^t$, then we have

$$\mathbf{g} - \mathbf{x}_o = \Delta\mathbf{x}^O - \Delta\mathbf{x}^t, \quad \mathbf{g}^O - \mathbf{x}_o = \Delta\mathbf{x}^t, \quad \mathbf{g}^t - \mathbf{x}_o = \Delta\mathbf{x}^O \quad (9)$$

Set $\mathbf{x}$ as the tool's trajectory in the world coordinates, and $\mathbf{x}^t$ and $\mathbf{x}^O$ as the decomposed trajectories of the tool and the object, then $\mathbf{x}$ always satisfies

$$\mathbf{x} - \mathbf{x}_o = k_1\left(\mathbf{x}^t - \mathbf{x}_o\right) + k_2\left(\mathbf{x}^O - \mathbf{x}_o\right), \quad (10)$$

where $k_1$ and $k_2$ are constant vectors. For the end point $\mathbf{g}^o$ and $\mathbf{g}^t$, (10) is still satisfied. Using (9), we can get

$$\left(I - k_1\right)\left(I + k_2\right)^{-1} = tr\left(\Delta\mathbf{x}^t\left(\Delta\mathbf{x}^O\right)^H\right), \quad (11)$$

which means that the factors $k_1$ and $k_2$ are determined by the vectors between the start and end contact points, and $I$ is an unit vector.

Using (10) and (11), we have

$$\mathbf{x} - k_1\mathbf{x}^t = k_2\mathbf{x}^O + \left(I - k_1 - k_2\right)\mathbf{x}_o. \quad (12)$$

Using DMP, the trajectory generated by the tool's contact point 2 in one-dimension is expressed as

$$\begin{cases} \tau\dot{v} = \alpha_z\left(\beta_z\left(g - x\right) - v\right) + f\left(s\right) \\ \tau\dot{x} = v \end{cases}. \quad (13)$$

Similarity, we assume the two trajectories (the pure tool's movements and virtual object movements) decomposed from $x$ are expressed by DMP as

$$\begin{cases}\tau\dot{v}^t = \alpha_z\left(\beta_z\left(g^t - x^t\right) - v^t\right) + f^t(s) \\ \tau\dot{x}^t = v^t\end{cases}, \quad (14)$$

$$\begin{cases}\tau\dot{v}^O = \alpha_z\left(\beta_z\left(g^O - x^O\right) - v^O\right) + f^O(s) \\ \tau\dot{x}^O = v^O\end{cases}, \quad (15)$$

where $x^t, v^t, g^t, x^O, v^O$ and $g^O$ are one-dimension values of vector $\mathbf{x}^t, \mathbf{v}^t, \mathbf{g}^t, \mathbf{x}^O, \mathbf{v}^O$ and $\mathbf{g}^O$, and $f^t(s)$ and $f^O(s)$ are forcing functions sharing the same phase variable $s$ with $f(s)$, and $\alpha_z$, $\beta_z$, and $\tau$ are common factors.

Using (13) and (14), we have

$$\begin{cases}\tau\dot{\tilde{v}}^t = \alpha_z\left(\beta_z\left(\tilde{g} - \tilde{x}^t\right) - \tilde{v}^t\right) + f(s) - k_1 f^t(s) \\ \tau\dot{\tilde{x}}^t = \tilde{v}^t\end{cases} \quad (16)$$

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

Taking (12) into (16), we have

$$\begin{aligned} k_2\tau\dot{v}^O = \alpha_z\left(\beta_z\left(\tilde{g}^t - k_2 x^O\right) - k_2 v^O\right) + f(s) - \\ k_1 f^t(s) + \alpha_z\beta_z\left(I - k_1 - k_2\right)x_o \end{aligned}. \quad (17)$$

Comparing (15) and (17), we can get

$$f(s) - k_1 f^t(s) - k_2 f^O(s) = 0 \quad (18)$$

It means that $f(s)$ is expressed by the linear composition of $f^t(s)$ and $f^O(s)$. As $k_1$ and $k_2$ are calculated by (11) after confirming the positions of the start point and end point, if a virtual trajectory $x^O$ based on $f^O(s)$ is generated, the pure tool's forcing function $f^t(s)$, as well as the operation skill $x^t$, can be directly calculated by (14). The SS skill learning and generalization can be realized through the following steps:

1. Acquire exact information of tool's shape, positions of the tool and object's contact points (as assumptions stated);
2. Use (11) to calculate $k_1$ and $k_2$;
3. Use (13) and (15) to learn the exact expression of the real tool's movement $x$ and virtual object movement $x^O$;
4. Use (18) to calculate $f^t(s)$ and get the pure SS skill (14);
5. Generalize the pure SS skill according to the new object's contact point and learn a new virtual object movement $x^O$ and a forcing function $f^O(s)$ according to the new tool's shape;
6. Use (18) to get $f(s)$ and use (13) to calculate the final tool's switching trajectory for a new tool and new object.

**Remark 2:** The SS skill can be generalized between different tools due to the essential conditions are merely contact points and the outline data of the tool and object that don't concern with other features. Contact point can be set autonomously by the visual recognition method or manually set by humans.

## IV. Experiments

The experiment is taken base on a small platform consisted of a Kinect for recording the object's movement and tool's positions, an Omni joystick for collecting position information and acting as an actuator for moving the object, and two self-made tools (a finger-shape tool and a rectangular slope tool) fixed at the end of the joystick (Fig. 7).

### *A. Setups and system modelling*

As the last three joints of Phantom Omni joystick are fixed and we add extra tools at the joystick's end, the kinematic and dynamics parameters of the joystick (Fig.7) are premeasured and recalculated.

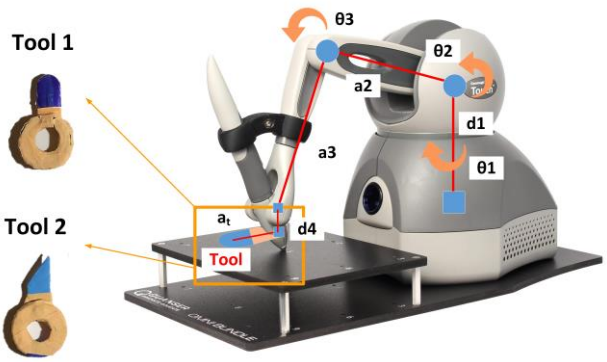



Figure. 7 Phantom Omni joystick and attached tool

DH parameters of phantom joystick is presented in Table I.

TABLE I. DH PARAMETER FOR THE PHANTOM AND TOOL

| **Joint** $i$ | $\alpha_i$ (mm) | $a_i$ (mm) | $d_i$ (mm) | $\theta_i$ |
|---|---|---|---|---|
| 1 | -pi/2 | 0 | 133.35 | $\theta_1$ |
| 2 | 0 | 133.35 | 0 | $\theta_2$ |
| 3 | 0 | 133.35 | 0 | $\theta_3$ |
| Tool | 0 | $a_t$ | 10 | 2pi/3 |

* $\alpha_i$ is link twist, $a_i$ is the link length, $d_i$ is the link offset, $\theta_i$ is the joint angle, and $a_t$ distance from robot end to the tool's center point

Parameters of the object and tools are measured and shown in the following Table II.

TABLE II. PARAMETER FOR OBJECT AND TOOLS

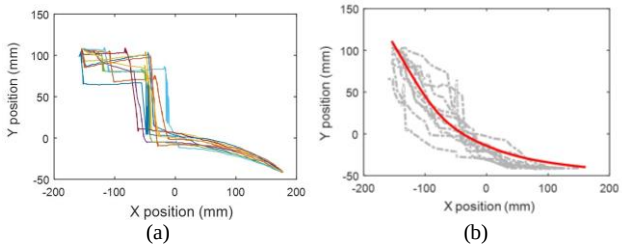
| **Object** | **Tool 1** | **Tool 2** |
|---|---|---|
| Radius =17.5mm | $a_i$=26mm; Tool's width = 12 mm; Radius of contact head =6mm; Left contact length = 10mm | $a_i$=28mm; Tool's width = 12mm; Left contact length = 5mm; Angle of inclined plane = pi/3 |

To enable the Omni to autonomously move as an actuator, the dynamics parameters of the joystick are estimated based on the method mentioned in [20].

### *B. Skill learning*

No matter OE skill or SS skill, the first step is to learn skills from demonstrations. OE skill is directly learned from object

trajectories (Fig.8 (a)). Like point A and B in Fig. 5 (b), we select some points at the trajectory corners as fixed points by hand, and build the constrained inequalities for the internal trajectory points. The constrained points will be recovered to the universal space following the procedure shown at the end of Section III. A. The recovered trajectories are colored in grey and shown in Fig.8 (b). Then a universal OE skill (red line in Fig.8 (b)) is learned from these trajectories by using DMP.

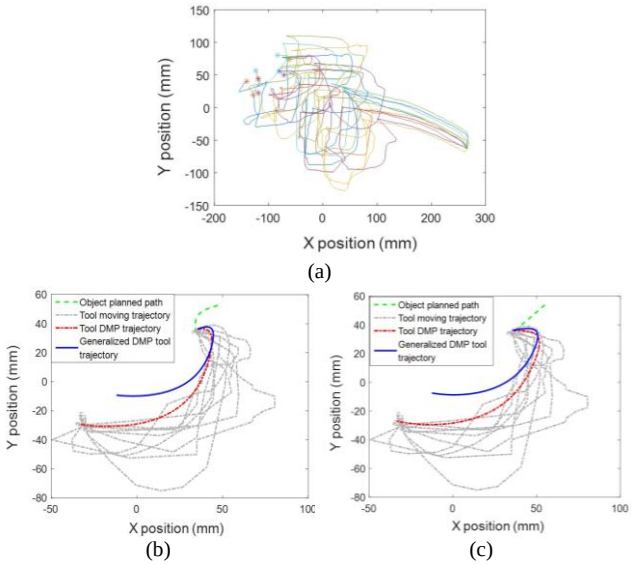



(a) (b)

Figure. 8 Object trajectory transformation and learning **(a)** original object trajecotries with constraints **(b)** recovered object trajectories (gray lines) and learned skill (red line) in the universal space

The results of SS skill learning are presented in Fig.9. First, we intercept the state switching actions from the original data presented in Fig.9 (a). Then the actions are aligned and shown as the grey lines in Figs.9 (b) and (c). By using DMP, the skills under the influence of the tool and the object are abstracted as the red dash line shown in Figs.9 (b) and (c).

Following (10), the trajectory point $x^t$ is calculated after setting the object virtual movements $x^o$. Here, we designed two virtual trajectories: an arching line and a direct line connecting the two contact points on the tool (green dash lines shown in Figs.9 (b) and (c)). Then, by using DMP, we can use (15) to fit the virtual curves and get $f^O(s)$. After calculating proper values of $k_1$ and $k_2$, $f^t(s)$ is achieved by (18) and the pure tool-use trajectories (blue solid lines shown in Figs.9 (b) and (c)) are also learned by (14). Comparing Fig.9 (b) and Fig.9 (c), the deviance degree between the blue curves and the start point in Fig.9 (b) is larger than the results in Fig.9 (c), and we can conclude that different virtual object movements will cause difference of the final learned pure skills.




(a)

(b) (c)

Figure. 9 Trajectories and DMPs-based tool's state switching skills. **(a)** tool's demonstrated movements **(b)** results of the arch-shaped object's virtual movements and learned SS skill **(c)** results of the line-shaped object's virtual movements and learned SS skill

### *C. Skill generalization*

The OE and SS skills are generalized for a different tool use case by setting different conditions in the environment. First, for the OE skill, the new tool, as Tool 2 in Fig. 7, has two functions: pushing the object leftward or pushing it diagonally with an angle of $0.75pi$ through an inclined plane. The start and end points are the same as in Fig.8(b). Compared to the demonstrations in Fig.5, we first generalize the skill learning in Fig. 8 (b) by the confirmed start and end points. Using the selected pushing points, we calculated the reachable areas of the new tool by its kinematic functions to generate new constraints of the tool in Cartesian space. In Fig.10, we show a feasible solution for the trajectory generalization, where the object is pushed leftward twice (red lines) and left forward (blue lines) twice till the end point.

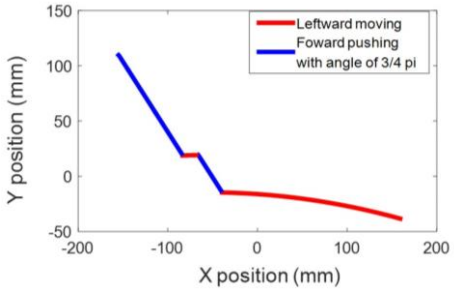
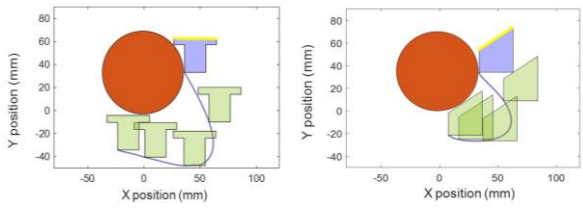



Figure. 10 OE skill generalization under the effect of a Trapezoid tool

The SS skill is generalized for two kinds of tool-use cases: A T-shape tool and a trapezoid-shape tool. The tools' initial locations are colored in blue in Fig. 11 (a) and Fig. 11 (b) and tools surfaces, highlighted in yellow, are prepared for the next contact. The contact points after state switching are randomly chosen along the yellow lines. The round objects are colored in dark red with the same size in Figs. 11 (a) and (b). Five internal steps are selected to present internal state to in Fig. 11.

During initialization, we confirm the starting contact positions and select the potential contact points on the object and the tool. Then, a virtual object motion is generated according to the steps presented at the end of Section III. B. Using the pure skill $x^t$ learned in Fig.8, we can generalize the SS skill changing along with the tool's shape.




(a) (b)

Figure. 11 Generalized tool's trajectories with different tool's shape and contact point changes **(a)** results of T-shape tool and the contact points locate at the right and bottom of the object **(b)** results of trapezoid-shape tool and the contact points locate at the right and bottom-right of the object

Seen from Fig. 11, the tools are moved to the desired positions and change the contact directions and positions finally. The generalized trajectories (blue lines in Fig. 11) are different seriously due to the tools' shapes.

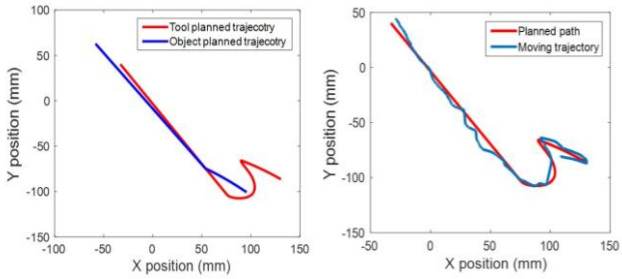
*D.Experiments based on integration of OE and SS skills*

The experiment is performed in four steps (as shown in Fig.3). First, an operator operates the joystick and controls it to move the object using tool 1. The Kinect and the joystick record the trajectory points of the center of the object and the end effector of the tool. After aligning the trajectories, compensating for platform deformation, and task segmentation for the generated trajectories, we create a skill learning dataset (Fig. 8 (a) and Fig. 9 (a)).

Second, the two skills are learned separately. For learning the OE skill, we select state change points (vertices) as fixed points and then the motions are recovered under the constraints caused by the tools and learned in the universal space (Fig. 8 (b)). Similarly, the SS skills are learned from the segmentations of the tool motions (gray lines in Figs. 9 (b) and (c)).

Third, the OE and SS skills are generalized sequentially. A new task should first contain information such as the start and end points, the outline of the tool and the contact points, the kinematic and dynamic parameters of the manipulator, and some special operational requirements such as obstacle avoidance or state change times. Then, the OE skill is generalized based on the known information, taking into account the physical conditions of the manipulator in Tables I and II and the object handling requirements (Fig. 10 (a)). The SS skills are then generalized based on the results of the OE skill generalization, which include exact start and end positions for each OE skill. (Fig. 11).

Finally, we can combine the OE and SS skills to generate the overall action of tool use triggered by a new tool for a new task by connecting the starts and ends of these skills. Fig. 12 (a) shows the generalized desired path for the object (blue) and the tool (red) and Fig. 12 (b) shows the actual movements of the tool recorded with the joystick..




(a) (b)

Figure. 12 Generalized object and tool's trajectories and the compare of the planned and real movements **(a)** Planned object and tool's trajectories **(b)** Compare of tool's planned path and real recording motions

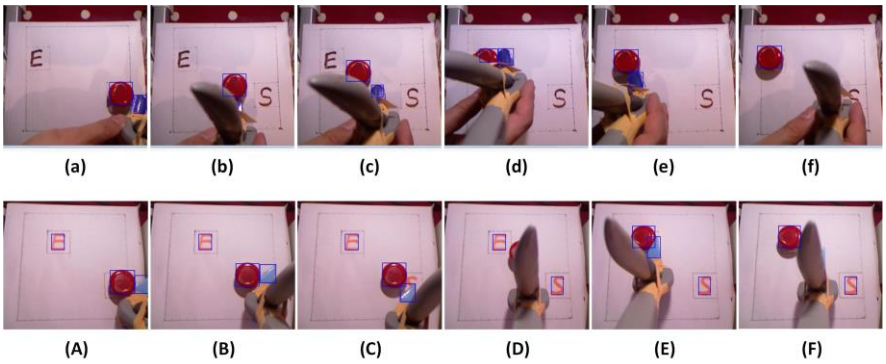



Figure. 13 Experiment taken based on the physical platform (a~f) demonstrations by human operator (A~F) the robot (joystick) completes new tasks (new tool and new map) autonamously based on learned skills. (a) and (A) are start state, (c)and (C) shows the states switching process, and (e) and (E) are the final reaching-target state, (f) and (F) are leaving screenshot of the tool

Fig.13 shows the experimental screenshots of the human demonstrations and the framework certification made by robot manipulating a new tool, and the object is pushed to the target location as expected.

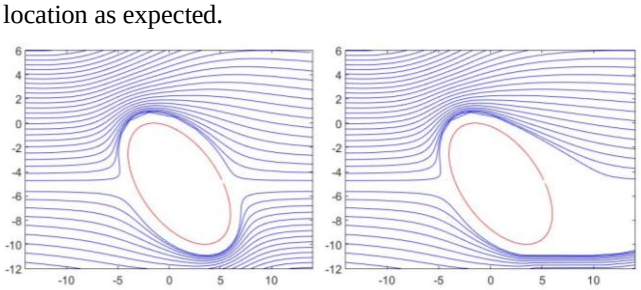


Figure. 14 Realize 'safety margin' and the long 'tail-effect' effect around the obstacle by (7)

Although we compensated for the actuator dynamics and used PD control for joystick position tracking, there is still some degree of position tracking error due to the stiffness of the joints and the obstacle forces generated by the object, as shown in Fig.12 (b).

*E. Extended functions of OE skill learning method*

Ref [19] proposed an idea of 'safety margin' to avoid confliction with obstacles, which it is easy to be realized in (7) by changing $r_g$ to $r_g + \delta$, and $\delta > 0$ represents the width of safety margin. As obstacle avoidance is a core application for DMP, compared with [19], two kinds of constraints caused by tools and the environmental conditions e.g. obstacles can be integrated in the same framework by using (7). Fig. 14 presents the preliminary results of realizing 'safety margin'

and ‘tail- effect’ effect around the obstacle, corresponding to the Figs.6 and 7 in [19].

### *F. Discussion*

In this paper, we conduct an exploratory study of skill learning and transfer between different tools and test the effectiveness of the proposed method through experiments. In this framework, we consider the constraints imposed by the environment and the shape of the tool. Some previous research has investigated skill learning and generalization technologies in a constrained environment, e.g., [10], [19]. There is limited research that considers both environment and tool influences and discusses skill transfer between different tools. However, the framework needs to be improved to reduce the manual settings, e.g., $x_o, x_b, k$ in (7), as well as the contact points of the tools, which can be determined by previous research in computer vision and deep learning, e.g., [12]. Due to the page limitations, the selection and optimization of these factors are not deeply discussed either.

The main contribution of this work is to provide a solution for tool manipulation capabilities after tool usage recognition, i.e., to answer the fourth question proposed by Brown and Sammut in [1]. We hope that the learned skills are reusable and can be applied to a new task after sufficient geometric information has been acquired and repeated demonstration has been avoided. The demos and experimental certifications will be performed on the 2D platform. For 3D space manipulation and 6D skills, there is still much to be done based on the current skill learning framework.

## V. CONCLUSION

For multi-tool use skills learning and transfer for object interaction, we propose a novel DMP-based framework that takes into account the effective working ranges of the tool, the selection of contact points, and the influence of the shapes of the tool and object. The OE and SS skills are learned from the tool and object trajectories using two improved DMP methods separately. By combining the two skills, the newly generated skill can be used to perform a task to reach an object using a new tool and considering the specific requirements.

An experiment about human interaction with the robot (Joystick) and autonomous task execution by the robot is conducted to verify the effectiveness of the proposed system. Due to the previous closure of the laboratory, the experiment was conducted only on a 2D platform. Future work will be based on real robots to perform 3D operations using standard tools. In addition, based on the work of this paper, some techniques, such as tool selection and effective region detection are expected to be added to the current framework to improve the robot's capabilities in using tools.

## REFERENCES


[1] S. Brown, and C. Sammut, "A relational approach to tool-use learning in robots, " in: International Conference on Inductive Logic Programming Springer, Berlin, Heidelberg. pp. 1-15, September, 2012.

[2] 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.

[3] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," Annual Review of Control, Robotics, and Autonomous Systems, no. 3, 2020.

[4] A. J. Ijspeert, Nakanishi J., and Schaal S., "Trajectory formation for imitation with nonlinear dynamical systems," in: Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2001. pp.752–757

[5] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," IEEE Trans. Robot., vol. 26, no. 5, pp.800-815, 2010.

[6] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," IEEE Trans. Robot., vol.28, no.1, pp.145-157, 2011

[7] A. Colomé, and C. Torras, "Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes", IEEE Trans. Robot., vol.34, no.3, pp.602-615, 2018.

[8] A. Montebelli, F. Steinmetz, and V. Kyrki, "On handing down our tools to robots: Single-phase kinesthetic teaching for dynamic in-contact tasks, " in: 2015 IEEE International Conference on Robotics and Automation (ICRA) , pp. 5628-5634, May, 2015.

[9] 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 Transactions on Mechatronics, vol. 23, no. 3, pp.1193-1203, 2018.

[10] Z. Lu, N. Wang and C. Yang, "A Constrained DMPs Framework for Robot Skills Learning and Generalization From Human Demonstrations," IEEE/ASME Transactions on Mechatronics, vol. 26, no. 6, pp. 3265-3275, Dec. 2021, doi: 10.1109/TMECH.2021.3057022.

[11] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, F. Li. and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision, " The International Journal of Robotics Research, vol. 39, no. 2-3, pp.202-216, 2020.

[12] Y. Bekiroglu, D. Song, L.Wang, and D.Kragic, "A probabilistic framework for task-oriented grasp stability assessment, " in: 2013 IEEE International Conference on Robotics and Automation, pp. 3040-3047, May. 2013.

[13] Z. Lu, N, Wang and C. Yang, "A novel iterative identification based on the optimised topology for common state monitoring in wireless sensor networks, " International Journal of Systems Science, vol. 53, no.1, pp. 25-39, 2022 .

[14] 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," in IEEE Transactions on Fuzzy Systems,

[15] Y. Zhu, Y. Zhao, and S. C. Zhu, "Understanding tools: Task-oriented object modeling, learning and recognition, " in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.2855-2864, 2015.

[16] R. Braud, A. Pitti and P. Gaussier, "A Modular Dynamic Sensorimotor Model for Affordances Learning, Sequences Planning, and Tool-Use," in *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 1, pp. 72-87, March 2018

[17] H. Hoffmann, Z. Chen, D. Earl, D. Mitchell, B. Salemi, and J. Sinapov, "Adaptive robotic tool use under variable grasps," *Robotics and Autonomous Systems*, no. 62, vol. 6, pp.833-846. 2014.

[18] N., Yamanobe, W., Wan, I.G., Ramirez-Alpizar, D., T., Petit, Tsuji, S., Akizuki, M., Hashimoto, K. Nagata, and K., Harada, A brief review of affordance in robotic manipulation research. *Advanced Robotics*, vol., no. 31(19-20), pp.1086-1101. 2017.

[19] S. M. Khansari-Zadeh, and A. Billard, "A dynamical system approach to realtime obstacle avoidance," Auton. Robots, vol.32, no.4, pp.433-454, 2012.

[20] T. Sansanayuth, I. Nilkhamhang, and K. Tungpimolrat, "Teleoperation with inverse dynamics control for phantom omni haptic device, " in 2012 Proceedings of SICE Annual Conference (SICE) , pp. 2121-2126, August, 2012,

[21] C., Lauretti, F. Cordella, and L. Zollo, "A Hybrid Joint/Cartesian DMP-Based Approach for Obstacle Avoidance of Anthropomorphic Assistive Robots". Int J of Soc Robotics vol 11, pp. 783–796, 2019.

[22] C. Zeng, Y. Li, J. Guo, Z. Huang, N. Wang and C. Yang, "A Unified Parametric Representation for Robotic Compliant Skills With Adaptation of Impedance and Force," in IEEE/ASME Transactions on Mechatronics, doi: 10.1109/TMECH.2021.3109160.