Arabian Journal for Science and Engineering
https://doi.org/10.1007/s13369-024-09592-4

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

A Code-Free Interactive Task Programming Interface for Robot Skill Construction

Ning Zhang[1] · Yongjia Zhao[1,2] · Shuling Dai[1]

Received: 6 April 2024 / Accepted: 9 September 2024

Abstract
To enhance production efficiency and quality, there is a rising interest in integrating robots into small manufacturing entities (SMEs) to enable flexible and agile production processes, thereby reducing redundancy. This poses challenges for robots as they must perform various tasks in unstructured environments without necessitating specialized programming training for workshop workers. This approach aims to reduce redundancy and streamline operations. To address these challenges, this paper presents a code-free system for programming robot tasks, which leverages an interactive programming interface and kinesthetic teaching methods. The system operates in three phases. Initially, the kinesthetic teaching method is employed to demonstrate the task, and the data generated from this demonstration are utilized for skill acquisition and visualization of the robot's state within the human–robot interaction software interface. Next, the demonstrated trajectories are learned through dynamic motion primitives, and various sub-skills form the skill library necessary for task completion. Ultimately, the skill library is activated to guide the actual robotic arm in executing the task. This approach allows end users to construct a skill library for a specific task without delving into task code-level programming. To validate the system's effectiveness, we invited experimenters to utilize our system for programming a designated task. The results demonstrate that users without programming experience can efficiently and flexibly employ our system to program robots for various tasks.

Keywords Human–robot interaction · Kinesthetic teaching · Learn from demonstration · Robot skills

1 Introduction

Over the past few years, we have entered the era of collaborative robots, where robots are no longer viewed as bulky machines confined to the production line but actively participate in shared workspaces alongside human workers. They can operate safely alongside humans without the need for fencing, enhance production processes, and facilitate the automation of novel processes.

To enhance both production efficiency and quality, there is a burgeoning interest in integrating robots into small manufacturing entities (SMEs) [1]. Nevertheless, SMEs encounter numerous distinct challenges. Initially, SMEs must possess the capability to swiftly adjust production lines for manufacturing new products in small batches. This necessitates robots to seamlessly execute diverse tasks within evolving workflows. Moreover, ideally, this repurposing should be achievable by shop floor workers without the need for extensive training or specialized programming skills. Lastly, systems must be adaptable to unstructured environments where equipment, tools, and components are often not fixed and may vary in position, orientation, or shape. Although low-cost collaborative industrial robots have witnessed increased adoption in robotic automation among SMEs, several fundamental issues persist as barriers hindering SMEs from implementing robotic automation in their factories.

A straightforward system that enables end users to intuitively program robots [2, 3] represents a crucial step toward transitioning robots from the laboratory to real-world applications. While experts can often effectively program robots

✉ Yongjia Zhao
zhaoyongjia@buaa.edu.cn

Ning Zhang
zy1703239@buaa.edu.cn

Shuling Dai
sldai@buaa.edu.cn

1 School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China

2 Jiangxi Research Institute, Beihang University, Nanchang 330096, Jiangxi, China

Published online: 19 September 2024

to execute complex tasks, such programming is typically knowledge-intensive, time-consuming, and often tailored to specific tasks. To tackle this challenge, recent efforts have concentrated on robotic learning by demonstration (LfD) [4–6] or programming by demonstration (PbD) [7–9]. In these approaches, non-expert users can teach robots task performance through demonstrations. Modern collaborative robotic arms like the UR series and KUKA iiwa7 support kinesthetic teaching [10, 11], where users can manually manipulate the end of the robotic arm to demonstrate task completion. These demonstrations eliminate the necessity for expertise in the robotic system and often only require a fraction of the time compared to manually designing a controller by an expert.

The interactive capabilities of current collaborative robotic arms enable them to demonstrate task trajectories through kinesthetic teaching [12–14]. However, they lack further semantic understanding and skill acquisition from the demonstration data, limiting their overall intelligence. With the aim of enhancing robotic arm intelligence, our goal is for the robotic arm to acquire skills and replicate the demonstrated data.

Ideally, learning from demonstration systems should be capable of mastering and generalizing complex tasks with minimal demonstrations, without the need for extensive robotic expertise. A significant portion of research in learning from demonstration (LfD) has concentrated on scenarios where a robot learns a comprehensive policy from a demonstration of a straightforward task with clearly defined starting and ending points. However, this approach frequently proves inadequate for complex tasks that cannot be effectively modeled using a single policy. Hence, structured demonstrations are frequently offered for a sequence of subtasks or skills that are simpler to learn and apply broadly than the entire task and that can be utilized across various tasks. However, manually segmenting tasks and providing demonstrations of individual skills present numerous challenges. As the most intuitive method of demonstrating a task is to execute it continuously from beginning to end, breaking down the task into component skills is not only time-consuming but also challenging. Effective segmentation demands an understanding of the robot's kinematics, internal representations, and existing skill sets. Defining skills requires qualitative judgment, as skills may be repeated within and across tasks. This includes determining when two parts can be considered a single skill and deciding on the suitable level of granularity for segmentation. Expecting users to manually manage these skill sets over time is impractical.

To achieve this goal, this paper introduces an interactive robot task programming system that relies on kinesthetic demonstration. With this system, operators lacking relevant robot programming experience can easily perform robot task programming. Initially, the kinesthetic teaching method is utilized to demonstrate the task, and the resulting data can be employed for skill acquisition and visualization of the robot's state within the human–computer interaction software interface. Next, the demonstration trajectories are acquired through dynamic motion primitives, and various sub-skills can form the skill repository needed to accomplish the task. Lastly, utilize the skill library to execute the task. As no code-level programming is necessary, the system outlined in this paper significantly reduces the requirements on operators.

In conclusion, the primary contributions of this paper can be summarized as follows:

(1) We introduce a framework that merges the advantages of kinesthetic demonstration and a visual programming interface, enabling intuitive teaching via demonstration and adaptable execution of structured tasks.
(2) We suggest a segmented skill demonstration and learning approach that links segmented demonstrations with skill repositories concurrently during a single kinesthetic demonstration.
(3) We showcase the entire system, offering experimental results to demonstrate the effectiveness of the proposed method in task teaching and execution.

2 Background and Related Work

2.1 End-User Robot Programming

End-user robot programming [15] is designed to enable users to program robots with complexity and ease of learning that matches their level of expertise. Previous research has investigated various programming modalities to enhance the usability of robot programming for end users These include natural language-based programming (e.g., [16, 17]), visual programming (e.g., [18, 19]), tangible programming (e.g., [20, 21]), and programming in augmented (e.g., [22]) or mixed reality (e.g., [23]).

Within the different end-user robot programming approaches, one direct approach for defining robot skills without manual programming is to demonstrate the task skill in question. Demonstrations can be supplied using kinesthetic teaching, where users physically guide the robot to perform the desired action (e.g., [28, 29]), through video (e.g., [24, 25]), or via teleoperation of the robot (e.g., [32, 33]). Robot motion trajectories can be demonstrated either as constant paths or as a collection of waypoints for the robot to follow.

To enhance the usefulness of human demonstrations, exploration in programming by demonstration (PbD) has explored methods to broaden demonstrations to diverse situations, allow robots to learn from numerous demonstrations of similar activities, and guide robots what actions to avoid. However, PbD is largely dependent on the quality of user-specified demonstrations. This work studies creating and revising tools aimed at assisting end-users in creating impactful and productive kinesthetic demonstrations. These demonstrations are intended to support later learning methods.

2.2 Robot Learning from Demonstration

Learning from demonstration (LfD) [26, 27] offers an intuitive approach to robot programming by allowing a human demonstrator to show the robot how to complete tasks. This method leverages the demonstrator's existing procedural knowledge, minimizing the need for specialized skills or training. LfD has gained significant attention due to its potential to simplify robot programming.

Robot imitation learning, a key aspect of LfD, involves reproducing the behavior demonstrated by a teacher. This is typically achieved by collecting and utilizing trajectories generated through kinesthetic teaching [28–31], telemanipulation [32–36], or virtual simulation environments [37, 38]. Kinesthetic teaching, where the robot is physically guided to exhibit desired behaviors, is particularly effective for learning and reproducing basic motor primitives. Our work aims to extend this by enabling robots to learn structured cooperative tasks from kinesthetic demonstrations.

For instance, Takano [39] developed a method to segment actions into dictionaries of basic movements, which can be combined to create complex behaviors. Similarly, Zuyuan Zhu et al. [26] proposed a method for learning grasping poses for assembly tasks through kinesthetic teaching and force controls. Their experiments on LEGO brick assembly demonstrated the feasibility of teaching robots to perform assembly tasks with simple demonstrations. However, further research is needed to evaluate the system's robustness across various assembly scenarios, such as sliding into grooves, screwing bolts, and chair assembly.

2.3 Trajectory Learning

Encoding skills at the trajectory level is a class of approaches to skill modeling that use variables in joint space, task space, or moment space to learn human motion. Among them, statistics-based methods have been widely used because they can effectively deal with the inherent variability of teaching actions, for example, there will be differences between teaching actions when the same person demonstrates the same task many times.

Trajectory learning [39–41] is a fundamental problem in robotics skill learning [42, 43]. It enables robots to learn useful skills and solve specific tasks. Many representations for encoding observed behaviors have been proposed for trajectory learning. Hidden Markov Models (HMMs) [45, 46] are a popular representation for trajectory learning that utilizes a Hidden Markov Model to encode the indicated trajectories, exploiting the concept of keypoints to generate generalized trajectories. Furthermore, David et al. proposed a Gaussian Mixture Model (GMM) for motion encoding, which has advantages over HMM during trajectory reconstruction [44].

In recent years, there have been many skill modeling studies using dynamic system-based approaches. The stability of the dynamic system can make the model robust to the disturbance of the environment. Dynamic Movement Primitives [47–52] (DMP) take another approach, combining nonlinear dynamical systems for trajectory modeling with statistical machine learning. Originally proposed by the team of Professor Stefan Schaal [47] in 2002, it is a method for trajectory imitation learning, which has been applied to various fields of robotics due to its highly nonlinear characteristics and high real-time performance. DMPs formulations have many desirable properties, such as stability and convergence, efficient coupling with other dynamical systems for trajectory modification, and robustness to environmental perturbations.

Work in DMP has looked at ways to generalize learned behavior and adapt it to new situations. For example, the literature [53] proposed the concept of query sub (Queries) combined with the DMP model. Its goal is not only to consider the motion control strategy learned from the teaching data (that is, the original DMP model parameters), but also to consider the robot's task recurrence. In the task parameters in the process, the DMP model that has been learned is linked with the current task state through the query. When the task status is different, it is very convenient to select/modify the model parameters to achieve skill generalization. They applied the proposed method to a robotic percussion instrument task. Reference [54] proposed an improved DMP model for robots to learn the skills of playing table tennis. Reinforcement learning has been applied to DMP to guide policy search and improve learning behavior [55–57]. The general process is: first, people teach the robot to get the teaching data; according to the teaching data, the DMP model is learned; in the skill reproduction stage, the learned DMP parameters are used as the initial strategy of the reinforcement learning algorithm, and the appropriate cost is defined according to the task requirements. The function optimizes the DMP model parameters until the task is completed. Literature [55, 56] proposed a weighted search strategy learning algorithm with reward, which is used for parameter adjustment and optimization of the DMP model, and the perception unit is coupled to the transformation system of each degree of freedom of the DMP model as an external environmental variable. Reference [57] applied the reinforcement learning algorithm to the cooperative arm grasping task of mobile robots, and optimized the high-dimensional (manipulator and manipulator) motion trajectories at the same time until the grasping task was successfully completed.

DMP is also widely used to demonstrate through learning. Qian Luo et al. [58] proposed a general handwriting learning system for a demonstration of a physical robot learning human handwriting to draw alphanumeric characters. DMP is also used in the industrial field. Reference [59] proposes a programming framework for exposure tasks based on demonstration learning. The framework is demonstrated on an industrial bonding task, showing that high-quality robotic behavior can be programmed from a single demonstration. Reference [60] describes an end-user instruction framework for industrial robotic assistants that supports complex, event-driven automation behaviors. The system has been deployed in laboratory experiments and real industrial tasks in an SME.

3 System Design

The architecture of the whole system is shown in Fig. 1. During the interactive demonstration of the task, the end user can program the robot and control the opening and closing of the end gripper through the interactive command menu of the human–robot interaction software. The demonstration of the task is achieved through the kinesthetic teaching function of the collaborative robot arm. Skill learning is based on dynamic motion primitives to encode demonstration trajectories to achieve trajectory-level learning.

The system of this paper mainly includes three parts: First, the Windows human–robot interaction software terminal, including the human–robot interaction UI menu and the digital twin simulation environment of the robot state visualization. The second is the skill learning and robotic arm control backend, including the DMP-based skill learning module, and the program package developed based on ROS and MoveIt to control UR5 and Robotiq hardware. The third is the hardware part, including robotic arms, grippers and experimental workbenches.

4 Human–Robot Interaction Programming Based on Kinesthetic Teaching

The system proposed in this paper supports human–robot interaction during task demonstration and task execution. In order to achieve natural interaction and incremental task learning, the system can transform demonstration and execution at any time. The system introduction is shown in Fig. 2. During the demonstration phase, the user can manually pull the robotic arm to perform a task correctly. For example, for the grasping task, the user can manually pull the robotic arm to the pre-grasp position of the object, and then control the closing of the gripper by clicking the “Close Gripper” button in the interactive system menu, thus completing the grasping task demonstration. The data throughout the teaching period have been recorded and supervised by the skill learning system. As the data input for skill learning, the segmented motion data are automatically associated with the corresponding subtasks. In this way, the low-level robotic actions that the user teaches through kinesthetic teaching are

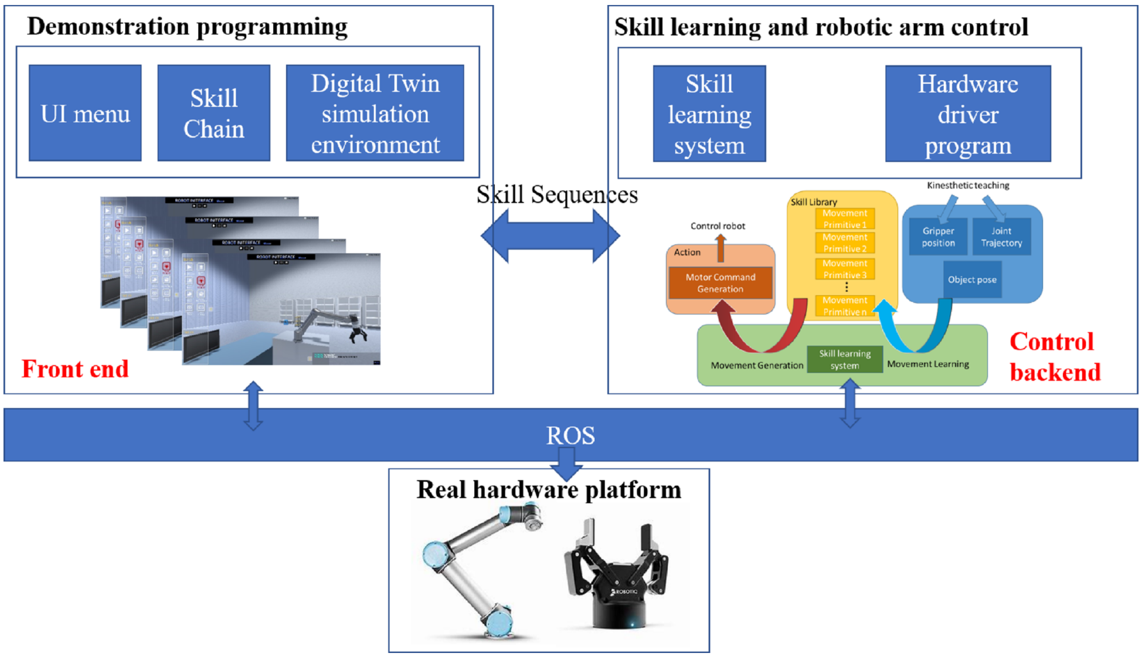
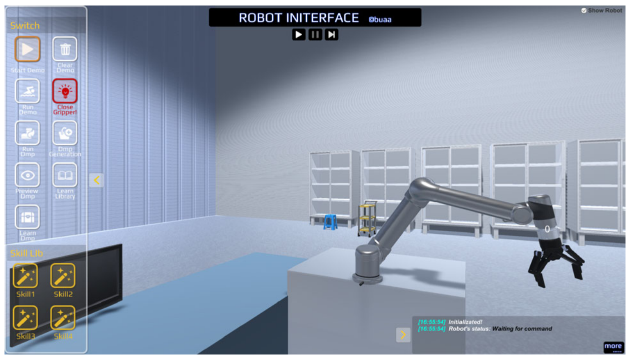


Fig. 1 System architecture



[16:55:54] Initializated!
[16:55:54] Robot's status: Waiting for command

Fig. 2 Human–robot interaction software

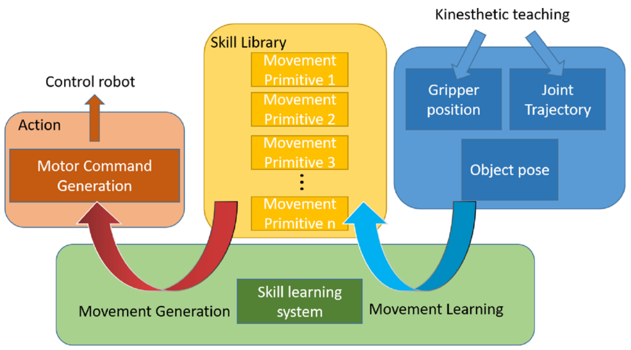


Fig. 3 Skill learning and reproduce

marked by the skill learning system subtask. The entire learning process will be described in further detail in the rest of this section.

4.1 Human–Robot Interaction System

Around the concept of end-user programming, we built an interactive system application where the end-user programs the robot. The interface of this application is shown in Fig. 2 below. The whole application is developed based on the unity game engine, which provides the function of demonstrating and learning the tasks of the robotic arm. On the left side of the application are some buttons for programming operations, which can be operated by clicking with the mouse. On the right is a model of UR5 collaborative robotic arm, a robotiq-2f 85 griper model, and a robotic arm simulation platform composed of some object models. Through the development kit based on ROS-Sharp, the unity application can communicate with the program on the ROS side. The robotic arm platform can be regarded as a simple digital twin system. After the system is started, the unity application subscribes to the joint position data of the real robotic arm platform in real time, and the robotic arm simulation platform updates in real time according to the subscribed data. The real hardware platform of the robotic arm achieves a fully synchronous update of the pose.

4.2 Human–Robot Skill Teaching

Through the human–robot interaction application on the Windows side and the kinesthetic teaching function of the collaborative robotic arm, we can complete the interactive demonstration and learning of robot tasks. Taking the task of grabbing a water bottle as an example, let us describe the entire flow of task programming.

(1) Start the programs on the Windows side and Ubuntu side, respectively
(2) Click the "Start Demo" button on the human–robot interaction software interface to start the expert demonstration
(3) Start the kinesthetic teaching mode of the UR5 robotic arm, and manually pull the end of the robotic arm to the pre-grab position of the water bottle
(4) Click the "End Demo" button on the human–robot interaction software interface to end the expert demonstration
(5) At this time, the trajectory data of the "approaching" water bottle is recorded by the skill learning system. Click the "Lean Dmp" button to invoke the DMP algorithm of the skill learning system to learn the "approaching" skill and store it in the skill library
(6) Click "Open Gripper" on the human–robot interaction software interface to complete the grabbing of the water bottle, and the skill learning system also records the grabbing action and stores it in the skill library
(7) Repeat the above process to complete the demonstration and learning of skills such as moving, releasing, and leaving, and finally form the "approach-grab-move-release-leave" skill library for the task of grabbing a water bottle. By clicking the "Dmp Generation" button, you can call the skill library of the water bottle grabbing task, click "Preview Dmp" to complete the trajectory planning, generate control commands for the robotic arm and gripper, and click the "Run Dmp" button to control the real robot platform to complete the task of grabbing the water bottle according to the planned trajectory.

4.3 Skill Learning

Through the human–robot interaction demonstration system, we will demonstrate each subtask of a given task separately and associate the subtask with the segmented demonstration data. To reproduce the actions of the robot, we encode the demonstration data of the subtasks into stable dynamic systems and refer to these systems as motion primitives. In this paper, the motion primitives are learned from demonstrations based on the DMP (Ijspeert et al. [48]) method. DMP encodes a motion primitive into a second-order nonlinear dynamic system. DMP models can generally be divided into discrete DMP models and rhythmic DMP models, respectively, for different types of motion: point-to-point motion and periodic motion. A single degree of freedom motion can be expressed by the following formula:

$$\begin{aligned}&\tau\dot{v}=K\left(x_g-x\right)-Dv+\left(x_g-x_0\right)f\left(s;\omega\right)\\&\tau\dot{x}=v\\&\tau\dot{s}=-\alpha_1 s\end{aligned}\tag{1}$$

For the sake of simplicity, the time variable is ignored here, for example, x_t means x. The formulation represents a transformation system that consists of two parts: a first-order spring-damper system and a nonlinear term, $f\left(s;\omega\right)$. In the formula, K and D represent the stiffness and damping of the system, respectively, and D is usually set as: $D=1/4K$. x_g and x_0 represent the target position and initial position of the motion, respectively; τ represent the time scaling constant, which is shared by all formulas; x and v represent the position and velocity of the motion trajectory, respectively, and the relationship between the two is shown in the formula; s represents the phase variable of the system, determined by a canonical system, see formula, where α_1 is a predefined constant. The nonlinear term $f\left(s;\omega\right)$ in the formula can be expressed by the following formula:

$$f\left(s;\omega\right)=\omega^T g\tag{2}$$

Among them, g represents a kernel vector, and w represents a set of policy parameters, which can directly determine the trajectory shape. The elements in the kernel vector are defined as:

$$[g]_n=\frac{\varphi_n\left(s\right)s}{\sum\limits_{n-1}^{N}\varphi_n\left(s\right)}\tag{3}$$

where $\varphi_n\left(s\right)$ represents a set of basis functions, and usually defines radial basis functions, namely Gaussian kernels:

$$\varphi_n\left(s\right)=\exp\left(-h_n\left(s-c_n\right)\right)\tag{4}$$

In the formula, c_n and h_n represent the center value and width value of the kernel function, respectively; N is the number of selected Gaussian models. Usually, c_n is uniformly distributed over the time length of the trajectory, h_n is selected according to experience; N can be selected according to the complexity of the task.

Generally, a supervised learning algorithm such as a locally weighted regression algorithm can be used to determine the model parameter w. Given a taught trajectory $x_t, \dot{x}_t, \ddot{x}_t$, where $t=[1\cdots T]$, $x_g=x\left(T\right)$, the target force function can be determined according to:

$$f_{t\arg et}=\frac{\tau\dot{v}+Dv-K\left(x_\partial-x\right)}{x_g-x}\tag{5}$$

Furthermore, w is determined by the following formula:

$$\min\sum\left(f_{t\arg et}-f\left(s\right)\right)^2\tag{6}$$

The above DMP model is for motion with one degree of freedom, while robotic skill learning usually involves multiple degrees of freedom, such as the need to control the end pose or joints of a robotic arm, which has multiple joints. In forming multi-joint systems from DMPs, we encode the movement trajectories of each joint as separate DMPs. These DMPs are then synchronized to ensure coordinated movement across all joints. The system adjusts the parameters of each DMP to account for the specific dynamics and constraints of the robotic arm, resulting in smooth and natural multi-joint motions. This approach allows for the flexible and adaptive generation of complex movements, even in unstructured environments.

5 Experiments

In this section, our system is applied to a real programming task to show that the proposed system can be used for fast and flexible programming of robotic tasks. By inviting some experimenters to evaluate the performance of learning and using our programming system.

5.1 Experimental Setup

In this section, we conducted some experiments to verify that our proposed method can (1) be used for fast robotic structured task programming and (2) transfer the acquired task knowledge in different task contexts. We consider two typical tasks: the block pick and place task and the water bottle pouring task.

Our experimental environment consists of both hardware and software components. The hardware part includes the UR5 cooperative manipulator, the Robotiq-2F85 gripper, and the experimental platform. As shown in Fig. 4, we use the Aruco marker to calibrate the pose transformation relationship between the camera and the manipulator. During the experiment, the marker is used to calculate the pose of the object and convert it to the base coordinate system of the manipulator.

The software part includes a computer running Windows 10, which is used to run the human–robot interaction software developed based on Unity, including the human–robot interaction menu and the digital twin environment of the robotic arm, as introduced in Section 4. Additionally, a program

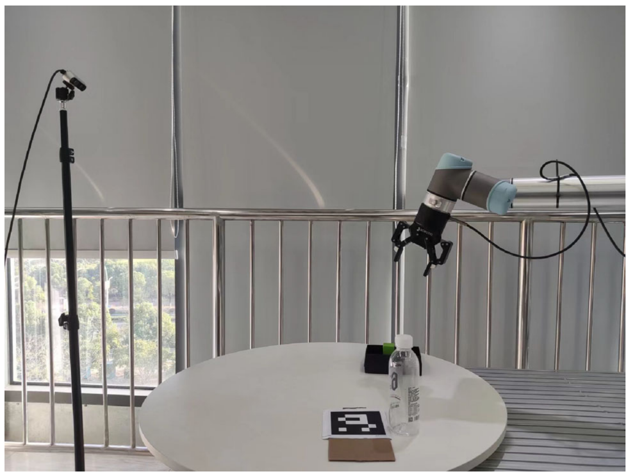


Fig. 4 Experimental scene

developed based on ROS includes the DMP skill learning algorithm and the robotic arm control algorithm. Communication between Unity and ROS is facilitated by the Ros-Sharp project, enabling direct communication with ROS topics, services, and actions from C# code.

5.2 Experimental Results and Discussion

5.2.1 Pick and Place Experiment

In the first experiment, we taught the robot how to pick and place green blocks into boxes. The first task contains five sub-skills: approach, grab, move, release, and leave. During the teaching process, the instructor controls the movement of the robot arm by simply moving the end of the robot arm and controls the opening and closing of the gripper and the skill learning of the movement trajectory through the Unity menu interface. Once the demonstration process is over, the robot has completed the learning of skills and can perform tasks by calling the learned skills library. In the demonstration process, each sub-skill is demonstrated separately and learned accordingly. The automatic segmentation of skills is done through the experience of human teaching experts, who teach a sub-skill at the beginning and end of the GUI interface. This forms a skill library, allowing the robot to complete the given task by sequentially executing the sub-skills. In the process of task execution, the skill library is called, and each sub-skill generates the motion control output of the corresponding robotic arm. A record of the task demonstration and robot execution is shown in Fig. 5.

In order to quantitatively evaluate the effectiveness of the proposed algorithm in this paper, we recorded the average time of 10 task teaching and execution and recorded the number of times the task was successfully executed under 10 cases. If the robot grabs the green block and moves the green block to a given position, it counts as a successful experiment. In order to demonstrate the robustness of our method to environmental changes, we change the initial position of the block, place the block at any position to carry out 10 experiments, and count the number of times the task is performed correctly to obtain the success rate. As shown in Table 1, the teaching of the block grab and place task takes about 50 s. The tasks were all successful in 10 experiments with a 100% success rate. Experimental results show that the proposed framework is able to transfer new skills to robotic devices in a fairly fast, natural, and efficient manner. In fact, in each experiment, the task was illustrated by a one-time kinesthetic teaching demonstration, with the help of the Unity menu interface for skill learning and control of gripper opening and closing. Also, note that the execution time of the task shown here is slightly longer than the time required for the demonstration of the task. A possible solution to reducing execution time is to increase the speed at which the robotic device executes actions.

In order to evaluate the effect of DMP-based skill learning of the system in this paper, we compared the demonstrated

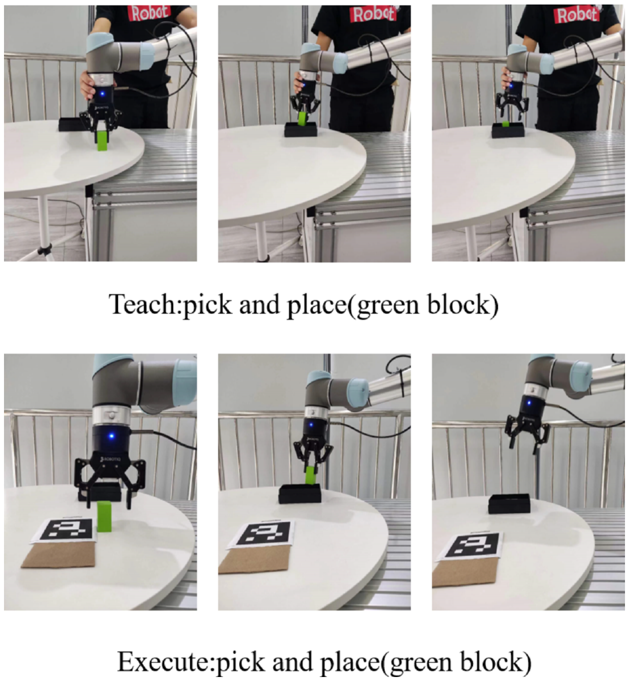

Teach:pick and place(green block)
Execute:pick and place(green block)

Fig. 5 The robot learns to pick and place tasks

Table 1 Results for ten repetitions of the pick and place green building block

	Pick block	Place block	Leave
Teaching time (s)	18.838	15.634	10.823
Execution time (s)	91.0969		
Success rate	100%		

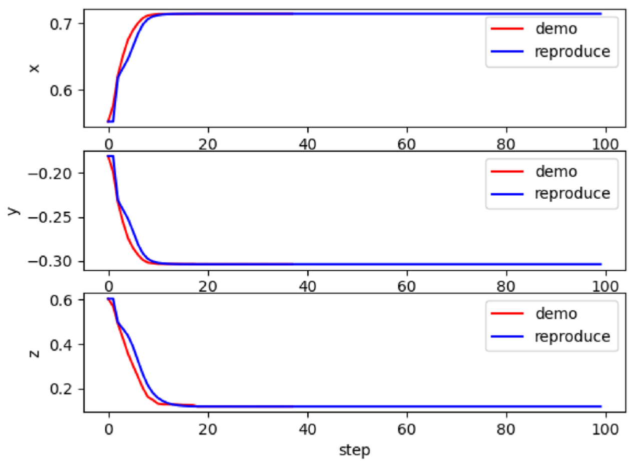


Fig. 6 Comparison of demonstration trajectories and generated trajectories in x, y, and z axes

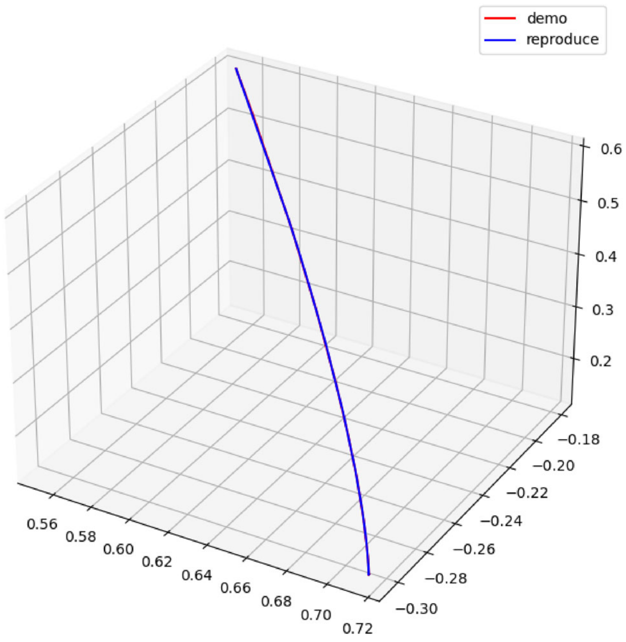


Fig. 7 Comparison of demonstration trajectory and generated trajectory in three - dimensional space

joint trajectory data with the trajectory generated by the DMP algorithm. Figure 6 is the comparison of the motion trajectory data in the x, y, and z axes, respectively, and Fig. 7 is the comparison of the motion data in the three-dimensional space, where the red line represents the motion trajectory curve collected during the teaching stage and the blue line is the trajectory learned by the DMP algorithm.

As can be seen from the experimental results in Figs. 6 and 7, the DMP trajectory learning algorithm is better able to imitate and demonstrate the trajectory, while ensuring the convergence of the starting point and the ending point.

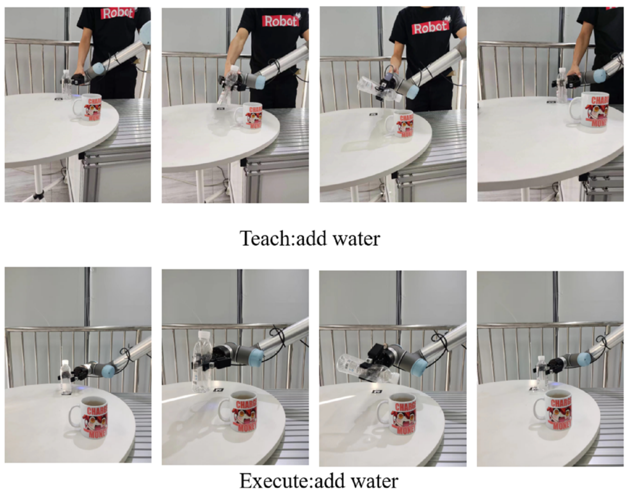


Fig. 8 The robot learns the task of grabbing a water bottle and pouring water

5.2.2 Water Bottle Pouring Experiment

This experiment demonstrates how a complex, structured task is learned and performed by the proposed framework. We set a water pouring task, and the robot must complete the following subtasks: (1) grab the water bottle; (2) move to the position of the water glass; (3) pour water into the water glass; (4) put the water bottle back to its position. As shown, we recorded pictures of the demonstration and robot execution of the four subtasks.

Similar to the previous experiment, we measured the teaching and execution time, and the success rate of 10 repetitions of the experiment, respectively, and the position of the water cup was randomly placed. The experimental results are shown in Table 2. On average, the total time for the demonstration water bottle pouring task is 55.3 s, and the time for the robot to perform the task is 130 s. Similar to the previous experiments, the longer execution time mainly depends on the speed limit set by the robot during mission planning. Table 2 also shows the demonstration times for each subtask. It can be seen from the experimental results that the water pouring subtask has the shortest time, because the water pouring subtask only needs to rotate the end joint of the robotic arm to complete the task demonstration.

In this experiment, we also noticed that the task success rate was very high (90%), with only 1 failure for 10 repetitions. In the failed experiment, the robotic arm poured water out of the water glass during the water pouring subtask, which was caused by inaccurate estimation of the water cup pose. A possible solution is to use more advanced object pose estimation methods that provide robustness to the system.

Table 2 Results for ten repetitions of the pouring water

	Take bottle	Pour water	Add water	Place bottle
Teaching time (s)	14.831	13.627	7.616	19.644
Execution time (s)	131.058			
Success rate	90%			

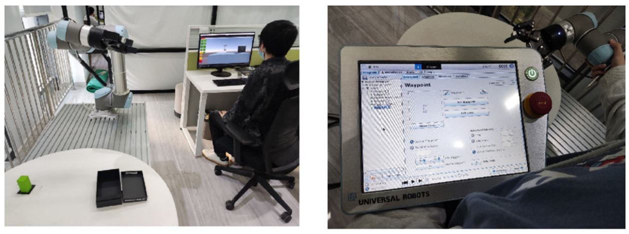

(a) Ours system (b) PolyScope

Fig. 9 Experimenters programmed the robot using our system and the PolyScope system separately

5.2.3 Comparative Experiment

We invite 10 experimenters to program a given task using our programming system and UR5's teach pendant, respectively, as shown in Fig. 9. Compare the experimental results of the two groups. Give the experimenters half an hour to learn our programming system and UR5 teaching pendant programming. For the robotic arm UR5 used in this article, the manufacturer developed the PolyScope system for collaborative robotic arm programming. The system has the functions of adding, previewing, and executing waypoints; path recording; and grabbing actions. These functions represent common programming methods for collaborative robotic arms. Compare the time and success rate of two groups of people programming the same task, and the experimental results are shown in Table 3.

It can be seen from the data in Table 3 that compared with the PolyScope system, our programming system can complete the programming of tasks in a shorter time. In terms of success rate, both experimenters achieved 100% success rate using our system, while the success rate on the PolyScope system was 75%. Both experimenters failed the first time using PolyScope. As can be seen from the programming time and success rate, our system is more conducive to programming by inexperienced users.

In addition, the teaching pendant programming system lacks the generalization of environmental changes. Once the environment changes, the robot needs to be reprogrammed to adapt to the environmental changes. Our system learns and stores skills, calls the skill library corresponding to the task to repeatedly execute the given task, and adapts to changes in the environment with the help of the sensory system. The sensing system in our system provides a camera to estimate the position of the operating object, and the trajectory generation algorithm can generate a trajectory according to the new position to adapt to the changes of the environment.

6 Conclusion and Future Work

The application of collaborative robotic arms greatly reduces the difficulty of deploying robotic arms in factories. The end user can complete the robotic arm task programming with the help of the teach pendant. But there are still some issues to be resolved. First of all, the end user needs a certain level of knowledge and time to learn the task programming through the teach pendant. Secondly, the programming of the teach pendant lacks generalization to environmental changes. Once the environment changes, the robot needs to be reprogrammed to adapt to the changes in the environment.

The interactive robot task programming system based on kinesthetic demonstration proposed in this paper solves the above two problems. End users can quickly program the robotic arm without any programming experience and can learn skills and adapt to changes in the environment. However, it is important to note that this technology has certain limitations. Specifically, the methodology is not suitable for scenarios that pose significant risks for direct demonstra-

Table 3 The experimental results of comparison between our system and PolyScope system

	Our system		PolyScope	
	Task programming time(s)	Success rate	Task programming time (s)	Success rate
Experimenter 1	55	100%	316	75%
Experimenter 2	65	100%	265	75%
Average time (s)	60		290.5	
Adapt to environmental change	Yes		No	

tion, such as those involving hazardous materials or extreme operating conditions. Additionally, the current implementation may not perform optimally at very high speeds due to the potential for increased error rates and decreased precision in trajectory execution. To address these limitations, future research could focus on enhancing the safety protocols for robot demonstrations in hazardous environments and optimizing the DMP algorithms for better performance at higher speeds.

Acknowledgements This work is funded by the National Key Research and Development Program of China (No. 2018AAA0102902).

Data availability The datasets generated or analyzed during this study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest No known conflict of interest.

References

1. Waurzyniak, P.: They're here: new collaborative robots lend a helping hand. Manuf. Eng. **150**(6), 49 (2013)
2. Krūmiņš, D.; Schumann, S.; Vunder, V.; Põlluäär, R.; Laht, K.; Raudmäe, R.; Kruusamäe, K.: Open remote web lab for learning robotics and ROS with physical and simulated robots in an authentic developer environment. IEEE Trans. Learn. Technol. (2024). https://doi.org/10.1109/TLT.2024.3381858
3. Myers, B.A.; Ko, A.J.; Burnett, M.M.: Invited research overview: end-user programming. In CHI'06 extended abstracts on Human factors in computing systems. pp. 75-80 (2006)
4. Argall, B.D.; Chernova, S.; Veloso, M.; Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**(5), 469–483 (2009). https://doi.org/10.1016/j.robot.2008.10.024
5. Billard, A.; Calinon, S.; Dillmann, R.; Schaal, S.: Survey: Robot programming by demonstration. Springer Handb. Robot. (2008). https://doi.org/10.1007/978-3-540-30301-5_60
6. Wu, H.; Yan, W.; Xu, Z.; Zhou, X.: A framework of improving human demonstration efficiency for goal-directed robot skill learning. IEEE Trans. Cogn. Dev. Syst. **14**(4), 1743–1754 (2021). https://doi.org/10.1109/TCDS.2021.3137262
7. Calinon, S.: Robot Programming by Demonstration. EPFL Press. (2009)
8. Calinon, S.: Learning from demonstration (programming by demonstration). Encycl. Robot. (2018). https://doi.org/10.1007/978-3-642-41610-1_27-1
9. Leiva, G.; Grønbæk, J.E.; Klokmose, C.N.; Nguyen, C.; Kazi, R.H.; Asente, P.: Rapido: prototyping interactive AR experiences through programming by demonstration. In The 34th Annual ACM Symposium on User Interface Software and Technology. pp. 626-637 (2021)
10. Meattini, R.; Chiaravalli, D.; Biagiotti, L.; Palli, G.; Melchiorri, C.: Combining unsupervised muscle co-contraction estimation with bio-feedback allows augmented kinesthetic teaching. IEEE Robot. Autom. Lett. **6**(4), 6180–6187 (2021). https://doi.org/10.1109/LRA.2021.3092269
11. Kurenkov, A.; Akgun, B.; Thomaz, A.L.: An evaluation of GUI and kinesthetic teaching methods for constrained-keyframe skills. In 2015 IEEE/RSJ International conference on intelligent robots and systems (IROS), pp. 3608–3613 (2015). https://doi.org/10.1109/IROS.2015.7353881
12. Eiband, T.; Liebl, J.; Willibald, C.; Lee, D.: Online task segmentation by merging symbolic and data-driven skill recognition during kinesthetic teaching. Robot. Auton. Syst. **162**, 104367 (2023). https://doi.org/10.1016/j.robot.2023.104367
13. Verheggen, J.; Baraka, K.: KRIS: A novel device for kinesthetic corrective feedback during robot motion. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 5041–5047 (2023). https://doi.org/10.1109/ICRA48891.2023.10160504
14. Ducaju, J.M.S.; Olofsson, B.; Robertsson, A.; Johansson, R.: Fast contact detection and classification for kinesthetic teaching in robots using only embedded sensors. In 2022 31st IEEE International conference on robot and human interactive communication (RO-MAN), pp. 1138–1145 (2022). https://doi.org/10.1109/RO-MAN53752.2022.9900800
15. Ajaykumar, G.; Steele, M.; Huang, C.M.: A survey on end-user robot programming. ACM Comput. Surv. **54**(8), 1–36 (2021). https://doi.org/10.1145/3466819
16. Buchina, N.; Kamel, S.; Barakova, E.: Design and evaluation of an end-user friendly tool for robot programming. In 2016 25th IEEE International symposium on robot and human interactive communication (RO-MAN). pp. 185-191 (2016). https://doi.org/10.1109/ROMAN.2016.7745109
17. Buchina, N.G.; Sterkenburg, P.; Lourens, T.; Barakova, E.I.: Natural language interface for programming sensory-enabled scenarios for human-robot interaction. In 2019 28th IEEE International Conference on robot and human interactive communication (RO-MAN). pp. 1-8 (2019). https://doi.org/10.1109/RO-MAN46459.2019.8956248
18. Coronado, E.; Mastrogiovanni, F.; Indurkhya, B.; Venture, G.: Visual programming environments for end-user development of intelligent and social robots, a systematic review. J. Comput. Lang. **58**, 100970 (2020). https://doi.org/10.1016/j.cola.2020.100970
19. Kuhail, M.A.; Farooq, S.; Hammad, R.; Bahja, M.: Characterizing visual programming approaches for end-user developers: a systematic review. IEEE Access **9**, 14181–14202 (2021). https://doi.org/10.1109/ACCESS.2021.3051043
20. Sefidgar, Y.S.; Agarwal, P.; Cakmak, M.: Situated tangible robot programming. In: Proceedings of the 2017 ACM/IEEE international conference on human–robot interaction, pp. 473-482 (2017, March). https://doi.org/10.1145/2909824.3020240
21. Kubota, A.; Peterson, E.I.; Rajendren, V.; Kress-Gazit, H.; Riek, L.D.: Jessie: Synthesizing social robot behaviors for personalized neurorehabilitation and beyond. In: Proceedings of the 2020 ACM/IEEE international conference on human–robot interaction, pp. 121-130 (2020, March). https://doi.org/10.1145/3319502.3374836
22. Yigitbas, E.; Jovanovikj, I.; Engels, G.: Simplifying robot programming using augmented reality and end-user development. In Human-Computer Interaction-INTERACT 2021: 18th IFIP TC 13 international conference, pp. 631-651 (2021). https://doi.org/10.1007/978-3-030-85623-6_36
23. Kapinus, M.; Materna, Z.; Bambušek, D.; Beran, V.: End-user robot programming case study: Augmented reality vs. teach pendant. In: Companion of the 2020 ACM/IEEE international conference on human-robot interaction, pp. 281-283 (2020). https://doi.org/10.1145/3371382.3378266
24. Qin, Y.; Wu, Y.H.; Liu, S.; Jiang, H.; Yang, R.; Fu, Y.; Wang, X.: Dexmv: Imitation learning for dexterous manipulation from human videos. In: European conference on computer vision, pp. 570–587 (2022, October). https://doi.org/10.1007/978-3-031-19842-7_33
25. Wang, C.; Fan, L.; Sun, J.; Zhang, R.; Fei-Fei, L.; Xu, D.; Anandkumar, A.: Mimicplay: Long-horizon imitation learning by watching

human play. (2023) arXiv preprint. https://doi.org/10.48550/arXiv.2006.04678 arXiv:2302.12422
26. Zhu, Z.; Hu, H.: Robot learning from demonstration in robotic assembly: a survey. Robotics **7**(2), 17 (2018). https://doi.org/10.3390/robotics7020017
27. Atkeson, C.G.; Schaal, S.: Robot learning from demonstration. In ICML, pp. 12-20 (1997).
28. Yu, C.; Yu, X.; Li, T.: Learning-model-based control for robot manipulators sensorless kinesthetic teaching using sparse feature dynamics. In: IEEE International conference on robotics and biomimetics (ROBIO), pp. 1-8 (2023). https://doi.org/10.1109/ROBIO58561.2023.10354598
29. Eiband, T.; Liebl, J.; Willibald, C.: Online task segmentation by merging symbolic and data-driven skill recognition during kinesthetic teaching. Robot. Auton. Syst. **162**, 104367 (2023). https://doi.org/10.1016/j.robot.2023.104367
30. Ajaykumar, G.; Stiber, M.; Huang, C.M.: Designing user-centric programming aids for kinesthetic teaching of collaborative robots. Robot. Auton. Syst. **145**, 103845 (2021). https://doi.org/10.1016/j.robot.2021.103845
31. Caccavale, R.; Saveriano, M.; Finzi, A.; Lee, D.: Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction. Auton. Robot. **43**, 1291–1307 (2019). https://doi.org/10.1007/s10514-018-9706-9
32. Zhang, T.; McCarthy, Z.; Jow, O.; Lee, D.; Chen, X.; Goldberg, K.; Abbeel, P.: Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In: 2018 IEEE International conference on robotics and automation (ICRA), pp. 5628–5635 (2018). https://doi.org/10.1109/ICRA.2018.8461249
33. Si, W.; Wang, N.; Yang, C.: A review on manipulation skill acquisition through teleoperation-based learning from demonstration. Cognitive Comput. Syst. **3**(1), 1–16 (2021). https://doi.org/10.1049/ccs2.12005
34. Hirschmanner, M.; Tsiourti, C.; Patten, T.; Vincze, M.: Virtual reality teleoperation of a humanoid robot using markerless human upper body pose imitation. In: 2019 IEEE-RAS 19th International conference on humanoid robots (humanoids), pp. 259-265 (2019). https://doi.org/10.1109/Humanoids43949.2019.9035064
35. Luo, J.; Liu, W.; Qi, W.; Hu, J.; Chen, J.; Yang, C.: A vision-based virtual fixture with robot learning for teleoperation. Robot. Auton. Syst. **164**, 104414 (2023). https://doi.org/10.1016/j.robot.2023.104414
36. Qin, Y.; Su, H.; Wang, X.: From one hand to multiple hands: imitation learning for dexterous manipulation from single-camera teleoperation. IEEE Robot. Autom. Lett. **7**(4), 10873–10881 (2022). https://doi.org/10.1109/LRA.2022.3196104
37. Stramandinoli, F.; Lore, K.G.; Peters, J.R.; O'Neill, P.C.; Nair, B.M.; Varma, R.; Reddy, K.K.: Robot learning from human demonstration in virtual reality. In: Proceedings of the 1st international workshop on virtual, augmented, and mixed reality for HRI (VAM-HRI). (2018).
38. Tagliabue, E.; Pore, A.; Dall'Alba, D.; Magnabosco, E.; Piccinelli, M.; Fiorini, P.: Soft tissue simulation environment to learn manipulation tasks in autonomous robotic surgery. In: 2020 IEEE/RSJ International conference on intelligent robots and systems (IROS), pp. 3261-3266 (2020). https://doi.org/10.1109/IROS45743.2020.9341710
39. Takano, W.; Nakamura, Y.: Real-time unsupervised segmentation of human whole-body motion and its application to humanoid robot acquisition of motion symbols. Robot. Auton. Syst. **75**, 260–272 (2016). https://doi.org/10.1016/j.robot.2015.09.021
40. Ahmadzadeh, S.R.; Kaushik, R.; Chernova, S.: Trajectory learning from demonstration with canal surfaces: A parameter-free approach. In: 2016 IEEE-RAS 16th International conference on humanoid robots (humanoids), pp. 544–549 (2016). https://doi.org/10.1109/HUMANOIDS.2016.7803328
41. Korkinof, D.; Demiris, Y.: Online quantum mixture regression for trajectory learning by demonstration. In: 2013 IEEE/RSJ International conference on intelligent robots and systems, pp. 3222–3229 (2013). https://doi.org/10.1109/IROS.2013.6696814
42. Tavassoli, M.; Katyara, S.; Pozzi, M.; Deshpande, N.; Caldwell, D.G.; Prattichizzo, D.: Learning skills from demonstrations: a trend from motion primitives to experience abstraction. IEEE Trans. Cogn. Dev. Syst. (2023). https://doi.org/10.1109/TCDS.2023.3296166
43. Kong, L.H.; He, W.; Chen, W.S.; Zhang, H.; Wang, Y.N.: Dynamic movement primitives based robot skills learning. Mach. Intell. Res. **20**(3), 396–407 (2023). https://doi.org/10.1007/s11633-022-1346-z
44. Duque, D.A.; Prieto, F.A.; Hoyos, J.G.: Trajectory generation for robotic assembly operations using learning by demonstration. Robot. Comput. Integr. Manuf. **57**, 292–302 (2019). https://doi.org/10.1016/j.rcim.2018.12.007
45. Lin, C.H.; Wang, K.J.; Tadesse, A.A.; Woldegiorgis, B.H.: Human–robot collaboration empowered by hidden semi-Markov model for operator behaviour prediction in a smart assembly system. J. Manuf. Syst. **62**, 317–333 (2022). https://doi.org/10.1016/j.jmsy.2021.12.001
46. Yan, J.; Huang, K.; Lindgren, K.; Bonaci, T.; Chizeck, H.J.: Continuous operator authentication for teleoperated systems using hidden Markov models. ACM Trans. Cybe. Phys. Syst. **6**(1), 1–25 (2022). https://doi.org/10.1145/3488901
47. Ijspeert, A.J.; Nakanishi, J.; Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings 2002 IEEE international conference on robotics and automation, pp. 1398–1403 (2002). https://doi.org/10.1109/ROBOT.2002.1014739
48. Ijspeert, A.J.; Nakanishi, J.; Hoffmann, H.; Pastor, P.; Schaal, S.: Dynamical movement primitives: learning attractor models for motor behaviors. Neural Comput. **25**(2), 328–373 (2013). https://doi.org/10.1162/NECO_a_00393
49. Saveriano, M.; Abu-Dakka, F.J.; Kramberger, A.; Peternel, L.: Dynamic movement primitives in robotics: a tutorial survey. Ind. Robot. **42**(13), 1133–1184 (2023). https://doi.org/10.1177/02783649231201196
50. Li, G.; Jin, Z.; Volpp, M.; Otto, F.; Lioutikov, R.; Neumann, G.: Prodmp: a unified perspective on dynamic and probabilistic movement primitives. IEEE Robot Autom Lett. **8**(4), 2325–2332 (2023). https://doi.org/10.1109/LRA.2023.3248443
51. Li, J.; Cong, M.; Liu, D.; Du, Y.: Enhanced task parameterized dynamic movement primitives by GMM to solve manipulation tasks. Robot. Intell. Autom. **43**(2), 85–95 (2023). https://doi.org/10.1108/RIA-07-2022-0199
52. Scheikl, P.M.; Schreiber, N.; Haas, C.; Freymuth, N.; Neumann, G.; Lioutikov, R.; Mathis-Ullrich, F.: Movement primitive diffusion: learning gentle robotic manipulation of deformable objects. IEEE Robot. Autom. Lett. (2024). https://doi.org/10.1109/LRA.2024.3382529
53. Ude, A.; Gams, A.; Asfour, T.; Morimoto, J.: Task-specific generalization of discrete and periodic dynamic movement primitives. IEEE Trans. Robot. **26**(5), 800–815 (2010). https://doi.org/10.1109/TRO.2010.2065430
54. Muelling, K.; Kober, J.; Peters, J.: Learning table tennis with a mixture of motor primitives. In 2010 10th IEEE-RAS International conference on humanoid robots, pp. 411–416 (2010). https://doi.org/10.1109/ICHR.2010.5686298
55. Kober, J.; Mohler, B.; Peters, J.: Learning perceptual coupling for motor primitives. In 2008 IEEE/RSJ International conference on intelligent robots and systems, pp. 834–839 (2008). https://doi.org/10.1109/IROS.2008.4650953

56. Kober, J.; Peters, J.: Policy search for motor primitives in robotics. Adv. Neural Inform. Process. Syst. (2008). https://doi.org/10.1007/s10994-010-5223-6
57. Li, Z.; Zhao, T.; Chen, F.; Hu, Y.; Su, C.Y.; Fukuda, T.: Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator. IEEE/ASME Trans. Mechatron. **23**(1), 121–131 (2017). https://doi.org/10.1109/TMECH.2017.2717461
58. Luo, Q.; Wu, J.; Gombolay, M.: A generalized robotic handwriting learning system based on dynamic movement primitives (dmps). (2020) arXiv preprint arXiv:2012.03898. https://doi.org/10.48550/arXiv.2012.03898
59. Iturrate, I.; Kramberger, A.; Sloth, C.: Quick setup of force-controlled industrial gluing tasks using learning from demonstration. Front. Robot. AI **8**, 767878 (2021). https://doi.org/10.3389/frobt.2021.767878
60. Guerin, K.R.; Lea, C.; Paxton, C.; Hager, G.D.: A framework for end-user instruction of a robot assistant for manufacturing. In 2015 IEEE international conference on robotics and automation (ICRA), pp. 6167-6174 (2015). https://doi.org/10.1109/ICRA.2015.7140065