Motion Planning for Human-Robot Collaboration based on Reinforcement Learning

Tian Yu, *Student Member, IEEE,* and Qing Chang, *Member, IEEE*

***Abstract*—Robots are good at performing repetitive tasks in modern manufacturing industries. However, robot motions are mostly planned and preprogramed with a notable lack of adaptivity to task changes. Even for slightly changed tasks, the whole system must be reprogrammed by robotics experts. Therefore, it is highly desirable to have a flexible motion planning method, with which robots can adapt to certain task changes in unstructured environments, such as production systems or warehouses, with little or no intervention needed from non-expert personnel. In this paper, we propose a user-guided motion planning algorithm in combination with reinforcement learning (RL) method to enable robots to automatically generate their motion plans for new tasks by learning from a few common tasks saved as primitive actions by kinesthetic human demonstrations. Features of these primitive actions are captured through screw transformation of the end-effector during the task. A mapping method is developed to convert features of primitive actions to new task segments and further used to construct the reward function in RL. A Q-learning algorithm is applied to train the motion planning policy, following which an adaptive motion plan for the new task can be generated or a request for additional primitive actions will be returned if current primitive actions are insufficient for satisfying new task constraints. Multiple experiments conducted on common tasks and scenarios demonstrate that the proposed RL-based motion planning method is effective.**

I. Introduction

In today's manufacturing systems, robot manipulators are usually required to finish repeatable tasks from a starting configuration to a goal configuration while maintaining specific constraints on positions and orientations during the task. However, when task instances change, reprogramming work for robots to satisfy new task constraints is time-consuming and incapable for non-expert human workers especially when the number of robots in human-robot-collaboration (HRC) is large. With the development of Industry 4.0, there's an increasing demand of robots working adaptively and smartly with humans in recent years [1]. Therefore, how to enable robots to collaborate with non-expert humans and automatically plan motions facing different tasks leave significant challenges to today's smart manufacturing.

To overcome these challenges, we involve both user demonstrations and robot motion planning to generate motion plans for tasks characterized by specific constraints. The use of user demonstration to teach a robot has been studied under the name of Learning from Demonstration (LfD). There is a substantial body of literature on LfD that can be classified based on the method of acquiring the demonstration such as using haptic gloves, image-based sensors, and kinesthetic demonstrations [2]. In this paper, we take advantage of kinesthetic demonstrations since there is no correspondence problem between the kinematic structure of the demonstrating system and the follower robot [3].

There have been a significant number of literatures on LfD that are useful for learning from kinesthetic demonstrations. Some researches use spline functions to decompose demonstrated trajectories to generate new trajectories for manipulation tasks [4]. However, they do not take care of noise in the demonstration which may be significant, especially when the motion information is obtained through human demonstrations. Some nonlinear regression techniques use statistical techniques to incorporate the uncertainty of sensing in the estimation. For instance, a Hidden Markov Model combined with Non-Uniform Rational B-Splines is used in [5] for trajectory approximation. A data-driven approach using Gaussian Mixture Model is adopted in [6]. However, these statistical approaches require multiple similar demonstrations for only one trajectories of the new task. This inefficient usage of user demonstrations would increase the labor of human operators in HRC. The dynamical systems based approach, or dynamical motion primitives (DMPs), on the other hand, can learn from single examples [7]. However, most works assume that there is a dynamical system modeling each degree-of-freedom (DoF) of the end-effector, which is not clear when both the position and orientation of the end-effector are relevant to the task. In addition, methods mentioned above are data-driven. The underlying kinetic transformations between each configuration in the user demonstration are not well utilized during the process of LfD.

For robot motion planning, existing algorithms can be divided into joint-space based approaches and task-space based approaches [8], [9]. For a robot manipulator, the joint space or configuration space is the set of all angles the joints can reach. The task space is the set of all end effector configurations of the robot manipulator. Joint space-based motion planning approaches have the advantage that they are (probabilistically or resolution) complete. Thus, in principle, they are guaranteed to give a feasible solution given enough time. However, in practice, they may not return a solution in reasonable time and a non-robotics expert will not be able to handle the situation. On the other hand, task-space based approaches handle task constraints more naturally, but they do

T. Yu is with Department of Mechanical and Aerospace Engineering, University of Virginia, Charlottesville, VA, 22903 USA (ty2yy@virginia.edu).

Q. Chang* is with Department of Mechanical and Aerospace Engineering and Department of Engineering Systems and Environment, University of Virginia, Charlottesville, VA, 22903 USA (qc9nq@virginia.edu)

* Corresponding author

not have any completeness guarantees in the presence of obstacles. Recently, Chakraborty et al. [10] develop a user-guided motion planning method in the task space that learns from only one human demonstration to generate motion plans for semantically similar task instances by using the imitated the human demonstration as a guidance given the new goal position. However, task constraints before the current pose blending into the imitated human demonstration are not guaranteed. In addition, the only explicit task constraint considered is the goal position. Other explicit task constraints like positions and orientations of some other configurations during the task are not considered, which makes it brittle to the change of task.

In this paper, to enable the robot to work with non-expert humans in certain HRC environment and reduce reprogramming when tasks change, we develop an RL-based motion planning method that can enable the robot to learn from one or multiple human demonstrated primitive actions to generate adaptive motion plans for new tasks in such HRC environment. First, primitive actions are recorded in the joint space through kinesthetic demonstrations based on some common tasks such as stacking a block, twisting a screw driver and filling a cup of water. Next, features of these primitive actions are captured by calculating screw transformations throughout demonstrations. Given new task constraints, we formulate the motion planning problem in the task space into a Markov Decision Process (MDP) framework and use the Q-learning method to train a motion planning policy to generate adaptive motion plans in the task space for the new task. Finally, inverse kinematics (IK) is used to calculate corresponding motion plans in the joint space to control the robot to execute learned motion plans.

The main contributions of this paper can be listed as: (1) Propose a method to abstract features of primitive actions; (2) Develop a method that can map features of primitive actions to the new task; (3) Formulate the robot manipulator learning from demonstration and motion planning problem as a reinforcement learning problem in the Markov Decision Process framework; (4) Propose a reasonable reward function based on the evaluation of the deviation between the generated motion plan and the task constraints.

The reminder of this paper is organized as follows: The mathematical preliminaries are introduced in Section II. The learning from demonstration and motion planning problem is stated in Section III. In Section IV, the learning from demonstration and motion planning problem is formulated as an MDP and solved by the Q-learning algorithm. Case studies and conclusions are provided in Section V and VI respectively.

II. Mathematical Background

In this paper, the joint space or configuration space is represented by $\mathcal{J}$, which is the set of all joint angles of the robot manipulator. $SE(3)$ denotes the Special Euclidean group of 3, which represents the task space contains all rigid body motions (i.e., rotations and translations) [11].

To describe the rigid body rotation and translation in $SE(3)$, a dual quaternion representation is adopted in this paper [12]. A dual quaternion $\boldsymbol{D}$ is defined as:

$$\boldsymbol{D} = \boldsymbol{d}_r + \frac{1}{2}\epsilon \boldsymbol{d}_t \otimes \boldsymbol{d}_r \tag{1}$$

where $\epsilon \neq 0$, but $\epsilon^2 = 0$. In this definition, $\boldsymbol{d}_t$ and $\boldsymbol{d}_r$ are two quaternions and $\otimes$ is the product of two quaternions. The translation of the rigid body is represented by the quaternion $\boldsymbol{d}_t$ which is denoted as:

$$\boldsymbol{d}_t = (0, \hat{\boldsymbol{t}}) \tag{2}$$

where $\hat{\boldsymbol{t}} = t_x\hat{\boldsymbol{i}} + t_y\hat{\boldsymbol{j}} + t_z\hat{\boldsymbol{k}}$ is the translation vector in $SE(3)$. The unit quaternion $\boldsymbol{d}_r$ representing the pure rotation of the rigid body can also be expressed as:

$$\boldsymbol{d}_r = \cos\left(\frac{\phi}{2}\right) + \hat{\boldsymbol{n}}\sin\left(\frac{\phi}{2}\right) \tag{3}$$

where $\hat{\boldsymbol{n}} = n_x\hat{\boldsymbol{i}} + n_y\hat{\boldsymbol{j}} + n_z\hat{\boldsymbol{k}}$ is a unit vector in $SE(3)$ representing the rotation axis, and ϕ is the rotation angle. Using this $\boldsymbol{d}_r$, any vector $\hat{v}$ can be rotated an angle ϕ about the axis $\hat{\boldsymbol{n}}$ by using the quaternion sandwich $\boldsymbol{d}_r\hat{v}\boldsymbol{d}_r^*$ [36], and $\boldsymbol{d}_r^*$ is the conjugate of $\boldsymbol{d}_r$. For more quaternion manipulations, we refer readers to [12].

III. Problem Formulation

In this paper, we consider a human-robot collaboration (HRC) environment, where some common tasks are demonstrated by human workers and recorded as primitive actions. Given constraints of new tasks and environment conditions, e.g., critical configurations of the end-effector and locations of an obstacle, a robot is required to finish these tasks by learning from primitive actions in such HRC environment. We assume that the robot has basic capability to move its end-effector from one configuration to another in the absence of any constraints. Our goal is to develop a method to enable the robot to generate adaptive motion plans for different new tasks to satisfy explicit and implicit task constraints.

A. Specify Manipulation Tasks for the Robot

In this paper, a manipulation task $\boldsymbol{TK}$ is defined as a sequence of n critical configurations based on the task constraints and environment conditions:

$$\boldsymbol{TK} = \{con_1, con_2, \dots, con_n\} \tag{4}$$

where $con_i,\ i = 1, \dots, n$, is a tuple of two $< \boldsymbol{P}_i, \boldsymbol{\theta}_i >$. In this tuple, $\boldsymbol{P}_i = [x_i, y_i, z_i]^T$ is a vector in $SE(3)$ that specifies the position of con_i, $\boldsymbol{\theta}_i = [\theta_{i1}, \theta_{i2}, \theta_{i3}]^T$, is a vector in $\mathcal{J}$ that defines Euler angles of con_i.

B. Features of Primitive Actions

Some common tasks in HRC environment, e.g., fastening screws can be regarded as primitive actions for a certain working scenario. A human can provide a kinesthetic demonstration of a primitive action by holding the arm of the robot and demonstrates a trajectory for accomplishing a given task instance. Throughout this demonstration, the explicit and implicit task constraints embedded in this human demonstrated primitive action can be recorded by many commercial robots such as UR robots in the joint space $\mathcal{J}$ as a time sequence of joint angles $\boldsymbol{\varphi}_{rec}$ as:

$$\boldsymbol{\varphi}_{rec} = \{\boldsymbol{\varphi}(1), \boldsymbol{\varphi}(2), \dots, \boldsymbol{\varphi}(m)\} \tag{5}$$

where each $\boldsymbol{\varphi}(t_{rec}) = [\varphi_1(t_{rec}), \varphi_2(t_{rec}), \dots, \varphi_r(t_{rec})]^T, t_{rec} = 1, 2, \dots, m$ represents joint angles of the manipulator, r is the DOF of the manipulator, and t_{rec} is the time sequence of the configurations reached during the motion. Using the forward kinematics mapping $\mathcal{FK}: \mathcal{J} \to SE(3)$, the corresponding human demonstrated primitive action in $SE(3)$ will be obtained as $\boldsymbol{DP} = \{\boldsymbol{D}_1, \boldsymbol{D}_2, \dots, \boldsymbol{D}_m\}$, in which each $\boldsymbol{D}_j$ is a dual quaternion denoted as $\boldsymbol{D}_j = \boldsymbol{d}_{rj} + \frac{1}{2}\epsilon \boldsymbol{d}_{tj} \otimes \boldsymbol{d}_{rj}, j = 1,2, \dots, m$.

Starting from $\boldsymbol{D}_1$, we compute the relative motion with respect to the final end effector configuration for each configuration $\boldsymbol{D}_j$. Using the dual quaternion representation, the transformation δ_j between the final configuration $\boldsymbol{D}_m$ and every other configuration $\boldsymbol{D}_j$ is:

$$\delta_j = \boldsymbol{D}_j^* \otimes \boldsymbol{D}_m \ , i = 1,2, \dots, m-1 \tag{6}$$

where $\boldsymbol{D}_j^*$ is the conjugate dual quaternion of $\boldsymbol{D}_j$ and $\otimes$ is the product of two dual quaternions. This δ_j is also a dual quaternion and can be written as: $\delta_j = \boldsymbol{d}_r^{\delta_j} + \frac{1}{2}\epsilon \boldsymbol{d}_t^{\delta_j} \otimes \boldsymbol{d}_r^{\delta_j}$. Since this transformation δ_j represents the relative translation and rotation of the end-effector between two configurations. It does not change with respect to the current configuration of the end-effector. In this way, all implicit task constraints are embedded in the sequence of δ_j during the motion, which is referred to as the feature of the human demonstration. As such, the feature of the k^{th} primitive action in the $SE(3)$ can then be described as:

$$\boldsymbol{HD}_k^{\delta} = \{\delta_1^{HD_k}, \delta_2^{HD_k}, \dots, \delta_{m-1}^{HD_k}\} \tag{7}$$

Suppose that there are h primitive actions in certain HRC environment, the set stores features of all h primitive actions can be denoted as:

$$\boldsymbol{LB} = \{\boldsymbol{HD}_1^{\delta}, \boldsymbol{HD}_2^{\delta}, \dots, \boldsymbol{HD}_h^{\delta}\} \tag{8}$$

Based on the definition of the manipulation task and features of primitive actions, the problem studied in this paper can be described as follows: *Given a set of primitive action's features* $\boldsymbol{LB} = \{\boldsymbol{HD}_1^{\delta}, \boldsymbol{HD}_2^{\delta}, \dots, \boldsymbol{HD}_h^{\delta}\}$ *and a new task* $\boldsymbol{TK} = \{con_1, con_2, \dots, con_n\}$ *in the task space* $SE(3)$*, develop a method to find motion plan* $\boldsymbol{MP}$ *in the joint space* $\mathcal{J}$ *for the new task by learning from human demonstrations in* $\boldsymbol{LB}$ *such that the explicit constraints specified in each* con_i *in* $\boldsymbol{TK}$ *is satisfied. If no* $\boldsymbol{MP}$ *can be found, then a request for additional human demonstrations is made to satisfy all task-relevant constraints in* $\boldsymbol{TK}$.

IV. Obtaining Motion Plans Trough Reinforcement Learning

In order to obtain the motion plan $\boldsymbol{MP}$ in $\mathcal{J}$ given a new task, we want to find out the motion plan $\boldsymbol{MP}$ in $SE(3)$ first since the new task and the set of human demonstrated features have been already established in $SE(3)$. After the motion plan $\boldsymbol{MP}$ in $SE(3)$ is generated, IK can be applied to convert the motion plan in $SE(3)$ to the motion plan in $\mathcal{J}$.

A. Map human demonstrated features to the new task

Since the new task $\boldsymbol{TK}$ is specified only on some critical configurations con_i, we want to develop a method that learns from human demonstrated primitive tasks to generate an intermediate motion plan between two critical configurations of the new task. The final motion plan $\boldsymbol{MP}$ in $SE(3)$ can consequently be obtained as a sequence of such intermediate motion plans.

First, let $\boldsymbol{tk}_s \subseteq \boldsymbol{TK}$ be a subset or the entire new task $\boldsymbol{TK}$ that can be defined as:

$$\boldsymbol{tk}_s = \{con_i, con_{i+1}, \dots, con_v\}, \ i \geq 1, i \leq v \leq n \tag{9}$$

Using the dual quaternion representation, the i^{th} configuration of the task $\boldsymbol{tk}_s$ can be written as:

$$\boldsymbol{D}_i^{tk_s} = \boldsymbol{d}_{ri}^{tk_s} + \frac{1}{2}\epsilon \boldsymbol{d}_{ti}^{tk_s} \otimes \boldsymbol{d}_{ri}^{tk_s}, i = 1,2, \dots, n \tag{10}$$

The transformation, $\delta_i^{tk_s}$, between the last critical configuration con_n and any other critical configuration con_i is defined as:

$$\delta_i^{tk_s} = \boldsymbol{D}_i^{tk_s^*} \otimes \boldsymbol{D}_n^{tk_s}, i = 1, \dots, n-1 \tag{11}$$

which can also be written as: $\delta_i^{tk_s} = \boldsymbol{d}_r^{\delta_i^{tk_s}} + \frac{1}{2}\epsilon \boldsymbol{d}_t^{\delta_i^{tk_s}} \otimes \boldsymbol{d}_r^{\delta_i^{tk_s}}$.

Therefore, features of the task $\boldsymbol{tk}_s$ can be represented as:

$$\boldsymbol{tk}_s^{\delta} = \{\delta_1^{tk_s}, \delta_2^{tk_s}, \dots, \delta_{n-1}^{tk_s}\} \tag{12}$$

Next, suppose that to finish this task segment $\boldsymbol{tk}_s$, the robot can learn from the feature $\boldsymbol{HD}_k^{\delta}$. We can define a mapping method $\boldsymbol{mp}_k^s: \boldsymbol{HD}_k^{\delta} \to \boldsymbol{tk}_s^{\delta}$, such that the relative transformation saved in $\boldsymbol{HD}_k^{\delta}$ can be scaled and fitted to the task constraints given in $\boldsymbol{tk}_s^{\delta}$. To obtain this $\boldsymbol{mp}_k^s$, the translation vector $\boldsymbol{d}_t^{\delta_j}$ of each δ_j in $\boldsymbol{HD}_k^{\delta}$ needs to be aligned to the vector $\boldsymbol{d}_t^{\delta_i^{tk_s}}$ of each $\delta_i^{tk_s}$ in $\boldsymbol{tk}_s^{\delta}$. The rotation from $\boldsymbol{d}_t^{\delta_j}$ to $\boldsymbol{d}_t^{\delta_i^{tk_s}}$ can be represented as a unit quaternion:

$$\boldsymbol{d}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s} = \cos\left(\frac{\phi_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s}}{2}\right) + \hat{\boldsymbol{n}}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s} \sin\left(\frac{\phi_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s}}{2}\right) \tag{13}$$

where $\hat{\boldsymbol{n}}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s} = \frac{\boldsymbol{d}_t^{\delta_j} \times \boldsymbol{d}_t^{\delta_i^{tk_s}}}{\left|\boldsymbol{d}_t^{\delta_j} \times \boldsymbol{d}_t^{\delta_i^{tk_s}}\right|}$ and $\phi_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s} = \cos^{-1}\frac{\boldsymbol{d}_t^{\delta_j} \cdot \boldsymbol{d}_t^{\delta_i^{tk_s}}}{\left|\boldsymbol{d}_t^{\delta_j}\right|\left|\boldsymbol{d}_t^{\delta_i^{tk_s}}\right|}$.

Therefore, each $\boldsymbol{d}_t^{\delta_j}$ of δ_j in $\boldsymbol{HD}_k^{\delta}, j = 1,2, \dots, m-1$ can be rotated using the quaternion sandwich $\boldsymbol{d}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s} \boldsymbol{d}_t^{\delta_j} \boldsymbol{d}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s^*}$. After scaling the vector $\boldsymbol{d}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s} \boldsymbol{d}_t^{\delta_j} \boldsymbol{d}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s^*}$ with $\frac{\left|\boldsymbol{d}_t^{\delta_1}\right|}{\left|\boldsymbol{d}_t^{\delta_1^{tk_s}}\right|}$. The final translation quaternion $\boldsymbol{d}_{tj}^{\boldsymbol{mp}_k^s}$ that maps the relative translation $\boldsymbol{d}_t^{\delta_i}$ in $\boldsymbol{HD}_k$ to $\boldsymbol{tk}_s$ can be obtained as:

$$\boldsymbol{d}_{tj}^{\boldsymbol{mp}_k^s} = \left(0, \frac{\left|\boldsymbol{d}_t^{\delta_1}\right|}{\left|\boldsymbol{d}_t^{\delta_1^{tk_s}}\right|} \boldsymbol{d}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s} \boldsymbol{d}_t^{\delta_j} \boldsymbol{d}_{\boldsymbol{HD}_k}^{\boldsymbol{tk}_s^*}\right), j = 1,2, \dots, m-1 \tag{14}$$

Since the detailed transformation between con_i and con_{i+1} are not specified, we can just use all the intermediate transformation in δ_j for the transformation between con_i and con_{i+1}. Let $\boldsymbol{d}_{rj}^{\boldsymbol{mp}_k^s} = \boldsymbol{d}_r^{\delta_j}, j = 1,2,\dots,m-1$, the mapping $\boldsymbol{mp}_k^s$ can finally be derived as:

$$\boldsymbol{mp}_k^s = \boldsymbol{d}_r^{\boldsymbol{mp}_k^s} + \frac{1}{2}\boldsymbol{d}_t^{\boldsymbol{mp}_k^s} \otimes \boldsymbol{d}_r^{\boldsymbol{mp}_k^s} \tag{15}$$

Let $\boldsymbol{TK} = \{tk_1, tk_2, \dots, tk_p\}$, the final motion plan $\boldsymbol{MP}$ in task space for $\boldsymbol{TK}$ can be determined by $\boldsymbol{MP} = \{\boldsymbol{mp}_{k_1}^1, \boldsymbol{mp}_{k_2}^2, \dots, \boldsymbol{mp}_{k_p}^p\}$, in which for each $q \in \{1,2,\dots,p\}$, $k_q \in \{1,2,\dots,h\}$. In the following subsections, we will discuss how to formulate the motion problem into a Markov Decision Process (MDP) and obtain this $\boldsymbol{MP}$ in $\boldsymbol{SE}(3)$ using an RL method.

B. MDP Formulation of the Problem

To find out the appropriate motion plan $\boldsymbol{MP}$ that satisfy all task constraints in $\boldsymbol{TK}$, one has to go through all subsets of $\boldsymbol{TK}$ and evaluate the corresponding motion plans generated by mapping each human demonstrated features in $\boldsymbol{LB}$ to $\boldsymbol{TK}$, which is a NP-hard problem [13]. Assume that there are x human demonstrations stored and y subsets of the new task TK, the computational complexity for evaluation of the motion plans by searching exhaustively is $O(x^y)$, which would be huge if the number of human demonstrations and the constraints of the new task is large. To solve this NP-hard problem, we formulate the problem into a MDP paradigm which is most common framework for RL that models the sequential decision making in uncertain environments [14].

Before we can apply RL algorithms to obtaining the ultimate motion planning policy π^*, we need to first properly define the three key components of MDP, which ares_t, a_t and r_t at time step t.

Given $\boldsymbol{LB} = \{\boldsymbol{HD}_1^\delta, \boldsymbol{HD}_2^\delta, \dots, \boldsymbol{HD}_h^\delta\}$ and $\boldsymbol{TK} = \{con_1, con_2, \dots, con_n\}$, the state s_t is defined as:

$$s_t = [\boldsymbol{CF}_t, \boldsymbol{tk}_t] \tag{16}$$

where $\boldsymbol{CF}_t$ is the current configuration of the end-effector at t, $\boldsymbol{tk}_t = \{con_j, con_{j+1}, \dots, con_n\}, j \in \{1,2,\dots,n\}$ is the subset of $\boldsymbol{TK}$ and $\boldsymbol{tk}_t' = \{con_j, con_{j+1}, \dots, con_k\}, k \le n$representing the current task the robot is going to satisfy.

The action a_t can be defined as:

$$a_t = [a_1(t), a_2(t), \dots, a_h(t)] \tag{17}$$

where each $a_i(t)$ is defined as:

$$a_i(t) = \begin{cases} k, & \text{if } \boldsymbol{HD}_i^\delta \text{ is mapped to } \boldsymbol{tk}_t' \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

which is to identify whether the human demonstrated feature $\boldsymbol{HD}_i^\delta$ is selected to be mapped to the current task $\boldsymbol{tk}_t'$.

To evaluate the action at t, the reward function r_t is defined as:

$$r_t = \begin{cases} -f\left(\delta_o, \delta_l^{\boldsymbol{tk}_t'}\right), & \forall \delta_l^{\boldsymbol{tk}_t'} \in \boldsymbol{tk}_t'^{\delta}, \exists \delta_o \in \boldsymbol{HD}_i^\delta \rightarrow \alpha\left(\delta_o, \delta_l^{\boldsymbol{tk}_t'}\right) \le \Delta_\alpha \\ -\infty, & \text{otherwise} \end{cases} \tag{19}$$

where $f\left(\delta_o, \delta_l^{\boldsymbol{tk}_t'}\right) = \sum_{l=j}^{k} \min_o \alpha\left(\delta_o, \delta_l^{\boldsymbol{tk}_t'}\right)$ and $\alpha\left(\delta_o, \delta_l^{\boldsymbol{tk}_t'}\right)$ is defined as:

$$\alpha\left(\delta_o, \delta_l^{\boldsymbol{tk}_t'}\right) = \min\left\{\left\|\boldsymbol{d}_r^{\delta_o} - \boldsymbol{d}_r^{\delta_l^{\boldsymbol{tk}_t'}}\right\|, \left\|\boldsymbol{d}_r^{\delta_o} + \boldsymbol{d}_r^{\delta_l^{\boldsymbol{tk}_t'}}\right\|\right\} \tag{20}$$

which is the Euclidean distance [15] between each δ_o in the feature $\boldsymbol{HD}_i^\delta$ and each feature $\delta_l^{\boldsymbol{tk}_t'}$ of the current task. Δ_α is a tolerance that is set to be 0.5 in this paper.

C. Applying Q-learning to Obtain the Optimal Motion Planning Policy

The basic idea of Q-learning is that we can define a function Q [14]:

$$Q(s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v(s', \pi) \tag{21}$$

such that $v(s,\pi^*) = \max_a Q^*(s,a)$. If we know $Q^*(s,a)$, then the optimal policy π^* can be found by simply identifying the action that maximizes $Q^*(s,a)$ under the state s. Starting with arbitrary initial values of $Q(s,a)$, the updating procedure of Q-learning is:

$$\begin{aligned} Q_{t+1}(s_t,a_t) = {} & (1-\alpha_t) Q_t(s_t,a_t) \\ & + \alpha_t \left[r_t + \gamma \max_a Q_t(s_{t+1}, a_t)\right] \end{aligned} \tag{22}$$

where $\alpha_t \in [0,1)$ is the learning rate and $\gamma \in (0,1)$ is the discount factor. The training process is shown in **Algorithm 1**. After the training, the ultimate policy π^* is determined as:

$$\pi^*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a' \in A(s)} \{Q(s,a')\} \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

where $A(s)$ is the set of all legal actions at state s in the Q-table $Q(s,a)$. The final motion plan $\boldsymbol{MP}$ is generated following the **Algorithm 2**.

Algorithm 1 Training of the RL-based Motion Planner

Input: $\boldsymbol{TK}$, $\boldsymbol{LB}$, ϵ, γ
Output: $Q(s,a)$
Initialize $Q(s,a)$ randomly
Initialize $t = 0$
Initialize s_0 with $\boldsymbol{CF}_0 = \boldsymbol{D}_1^{tk_1}$ and $\boldsymbol{tk}_0 = \boldsymbol{TK}$
For $episode = 0,1,\dots,100$ **do**
  While the last con_n of $\boldsymbol{TK}$ is not reached **do**
    Choose a_t=arg $\max_{a \in A(s_t)} Q(s_t,a)$using policy derived from $Q(s,a)$ (e.g. ϵ-greedy)
    Take action a_t, observe r_t
    Update $Q_{t+1}(s_t,a_t)$ based on Eqn. (22)
  End While
End For
Output $Q(s,a)$

Algorithm 2 Generating Motion Plan in $\boldsymbol{SE}(3)$

Input: $\boldsymbol{TK}$, $\boldsymbol{LB}$, $\epsilon, \gamma, Q(s,a)$
Output: $\boldsymbol{MP}$

Initialize s_0 with $\boldsymbol{CF}_0 = \boldsymbol{D}_1^{tk_1}$ and $\boldsymbol{tk}_0 = \boldsymbol{TK}$

For $t = 0,1,\dots,T$ **do**

Find legal action list $A(s_t)$ from $Q(s_t, a) \rightarrow a \in A(s_t)$

Find the optimal action as $a_t = \arg\max_{a \in A(s_t)} Q(s_t, a)$

Map $\boldsymbol{HD}_i$ to $\boldsymbol{tk}_t'$ according to a_t

$MP_t \leftarrow \boldsymbol{D}_n^{tk_t'} \otimes \boldsymbol{mp}_i^{t*}$

End For

$\boldsymbol{MP} \leftarrow \{MP_0, MP_1, \dots, MP_T\}$

Output $\boldsymbol{MP}$

V. Case Study

In order to validate effectiveness of the proposed method in generating motion plans for new tasks, multiple experiments are conducted on the UR5e platform. In this case study, two performance metrices are considered: (1) The accumulated reward of the motion plan in $SE(3)$; (2) The successful rate of applying the motion plan in $\mathcal{J}$ for new tasks. From the case study, two significant results can be concluded: (1) The proposed method is effective in combining different features of human demonstrations to generate motion plans for the new task; (2) The proposed method is effective in requesting additional human demonstrations if no features of primitive actions are semantically similar to the new task.

A. Record Primitive actions in $\mathcal{J}$

As shown in Fig. 1, three primitive actions are recorded in $\mathcal{J}$ through kinesthetic demonstrations, including one screwing task, one filling task and one stacking task.

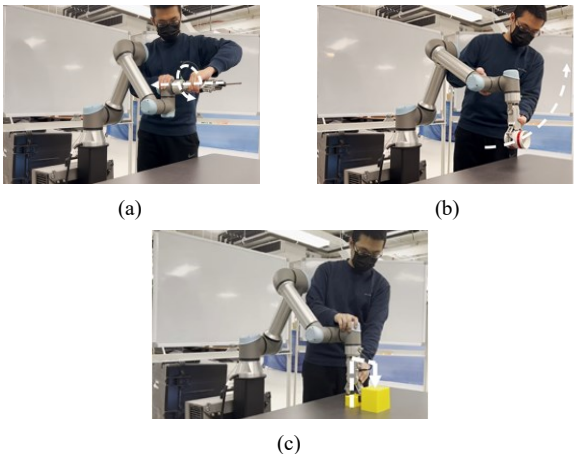

Figure 1. Kinesthetic demonstrations of 3 most common tasks. (a) Screwing task. (b) Filling task. (c) Stacking task.

B. Training the RL-based Motion Planner in SE(3)

In order to obtain a general motion planning policy in $SE(3)$ for the HRC scenario, we implement Algorithm 1 to train a Q-table initialized with random Q values in Matlab using a 4-core 4.0GHz Intel Core i7 processor. 20 new tasks with four critical configurations of each that cover all features are used during the training. Positions of these critical configurations are randomly generated within a $50 \times 50 \times 50\ cm^3$ workspace. Corresponding Euler angles of each critical configuration are also randomly selected from a set $\{-\pi, -\pi/2, 0, \pi/2, \pi\}$. The total training episode for each new task is set to be 100. The total computation time is 1927.42 seconds.

To monitor the training process, accumulated rewards for each new task are recorded every two iterations. The average accumulated reward for all 20 new tasks is shown in Fig. 2. Although the training rewards are noisy before 50 episodes, the underlying trend is that the rewards are increasing with training episodes. It can be observed that the reward reaches a steady level after around 50 episodes. This indicates that a steady motion planning policy in $SE(3)$ that can map appropriate features of human demonstrations to new tasks with semantically similar features is generated for the assemble and loading/unloading scenario.

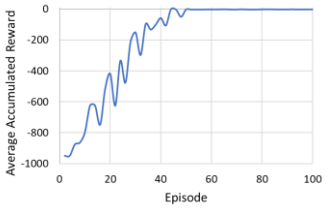

Figure 2. Training process for motion plans in $SE(3)$

C. Evaluation of the Trained Motion Planning Policy

To evaluate the performance of the trained motion plan policy in $SE(3)$ for the HRC scenario, two new tasks, namely, a filling-and-pouring task and a passing task, are used as examples to demonstrate the method. The trained Q-table from is used as the input to Algorithm 2 to generate motion plans for new unseen tasks in $SE(3)$. IK is used to calculate the final motion plan in $\mathcal{J}$. For each task, 20 experiment trials are conducted. For each trial, if the final motion plan in $\mathcal{J}$ is collision-free and satisfy all task constraints, such trial would be regarded as a successful one. The successful rate for each new task is recorded to evaluate the motion planning policy.

Filling-and-Pouring Task: In this task, the end-effector is required to fill water to the blue cup (shown in Fig. 3), then reach a goal position above the white cup (shown in Fig. 3), and finally pour water to the white cup. The location of the white cup is on the surface of a desk within a workspace of $20 \times 20\ cm^2$. We can specify 4 critical configurations based on the described task. Sample critical configurations from con_1 to con_4 are presented in Table 8, where $-\pi \le \gamma \le \pi$, $-0.5 \le x \le 0.7$, $-0.2 \le y \le 0$ for different trials. For 20 experiment trials, a successful rate of 80% is observed. We use the final motion plan in $\mathcal{J}$ for one trial as an example as shown in Fig. 3 to illustrate the result. It is noticed that the features of 3 human demonstrated tasks, namely filling, stacking, and screwing, are learned and mapped to the segment between con_1 and con_2, the segment between con_2 and con_3, and the segment between con_3 and con_4, respectively. In this experiment, the proposed method can identify and compose the appropriate features of primitive actions to perform a new task.

TABLE I. Critical Configurations of the Filling-and-Pouring Task

	con_1	con_2	con_3	con_4
$\boldsymbol{P}$	(-0.4,-0.1,0)	(-0.5,0.1,0.1)	$(x, y, 0.1)$	$(x, y, 0)$
$\boldsymbol{\theta}$	$(0,-\pi,0)$	$(0, -\pi/2,\gamma)$	$(0, -\pi/2,\gamma)$	$(\pi/2,0, -\pi/2)$

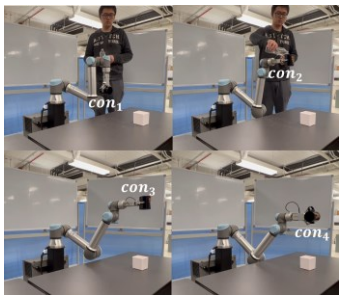

Figure 3. Final motion plan for the filling-and-pouring task.

Passing Task: In this task, the end-effector is required to pass the screw driver to the human in a specific orientation. By applying the same trained general motion plan policy, a zero successful rate is observed in the experiment, which indicates additional human demonstrations are needed. A closer examination reveals that none of the three features of primitive actions is semantically similar to the feature of the task segment between con_2 and con_3, which requires a 90-degree rotation about its body-fixed x-axis clockwise. Therefore, additional human demonstration is requested for this feature.

TABLE II. CRITICAL CONFIGURATIONS OF THE PASSING TASK

	con_1	con_2	con_3
$\boldsymbol{P}$	(-0.5,-0.1,0.2)	(-0.5,0.1,0.4)	(-0.7, 0.2, 0.4)
$\boldsymbol{\theta}$	$(\pi,0,\pi)$	$(\pi/2, 0,-\pi)$	$(-\pi/2, 0,-\pi/2)$

With this additional human demonstrated primitive action shown in Fig. 4 (a) added, the motion planning policy is retrained using Algorithm 1 with the Q-table trained in section 6.2. As shown in Fig. 4 (b), the accumulated reward reaches a steady value after around only 8 iterations. Then by applying the newly trained policy, the motion plan is generated as shown in Fig. 4 (c). All 20 trials are witnessed successful. The result shows that the features of the human demonstrated screwing task is learned and mapped to the task segment between con_1 and con_2. The feature of the newly added human demonstration is learned and mapped to the task segment between con_2 and con_3.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a method for incorporating human demonstrations in RL-based motion planning. By learning from human demonstrated features of primitive actions, the task-space RL-based motion planner can effectively generate motion plans for new tasks or request additional human demonstrations for tasks if the features of all primitive actions are insufficient to finish the task. Using inverse kinematics (IK), motion plans in the joint space can be obtained and successful trials are able to be achieved in real-world experiments. In future work, we plan to extend our proposed method to mobile manipulators. We also plan to integrate with the human motion perception, recognition, and prediction for realistic implementation.

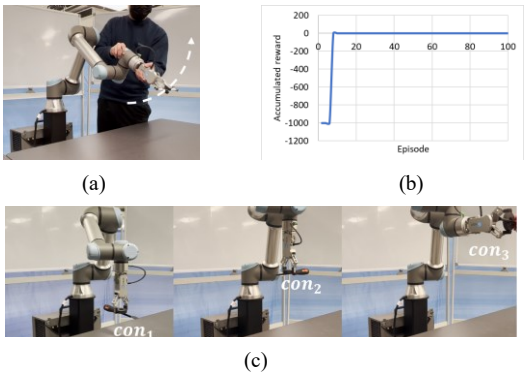

Figure 4. Outline for the passing task. (a) Additional human demonstration. (b) Training process with newly added human demonstration. (c) Final motion plan for the pass task.

ACKNOWLEDGMENT

This work was supported by the U.S. National Science Foundation (NSF) Grant. CMMI1853454.

REFERENCES

[1] S. El Zaatari, M. Marei, W. Li, and Z. Usman, "Cobot programming for collaborative industrial tasks: An overview," *Rob. Auton. Syst.*, vol. 116, pp. 162–180, 2019.

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

[3] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, 2018.

[4] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, 2003, vol. 2, pp. 1444–1449.

[5] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Rob. Auton. Syst.*, vol. 54, no. 5, pp. 409–413, 2006.

[6] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 37, no. 2, pp. 286–298, 2007.

[7] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.

[8] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[9] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Robot. Autom.*, vol. 3, no. 1, pp. 43–53, 1987.

[10] R. Laha, A. Rao, L. F. C. Figueredo, Q. Chang, S. Haddadin, and N. Chakraborty, "Point-to-point path planning based on user guidance and screw linear interpolation," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2021, vol. 85451, p. V08BT08A010.

[11] J. M. Selig, *Geometric fundamentals of robotics*, vol. 128. Springer, 2005.

[12] L. F. da C. Figueredo, "Kinematic control based on dual quaternion algebra and its application to robot manipulators," 2016.

[13] D. P. Bovet, P. Crescenzi, and D. Bovet, *Introduction to the Theory of Complexity*, vol. 7. Prentice Hall London, 1994.

[14] D. Silver, "Reinforcement Learning and Simulation-Based Search," 2009.

[15] L. Kavan, S. Collins, C. O'Sullivan, and J. Zara, "Dual quaternions for rigid transformation blending," *Trinity Coll. Dublin, Tech. Rep. TCD-CS-2006-46*, 2006.