Logic Learning from Demonstrations for Multi-step Manipulation Tasks in Dynamic Environments

Yan Zhang, Teng Xue*, Amirreza Razmjoo*, Sylvain Calinon

***Abstract*—Learning from Demonstration (LfD) stands as an efficient framework for imparting human-like skills to robots. Nevertheless, designing an LfD framework capable of seamlessly imitating, generalizing, and reacting to disturbances for long-horizon manipulation tasks in dynamic environments remains a challenge. To tackle this challenge, we present Logic-LfD, which combines Task and Motion Planning (TAMP) with an optimal control formulation of Dynamic Movement Primitives (DMP), allowing us to incorporate motion-level via-point specifications and to handle task-level variations or disturbances in dynamic environments. We conduct a comparative analysis of our proposed approach against several baselines, evaluating its generalization ability and reactivity across three long-horizon manipulation tasks. Our experiment demonstrates the fast generalization and reactivity of Logic-LfD for handling task-level variants and disturbances in long-horizon manipulation tasks.**

Project webpage: https://sites.google.com/view/logic-lfd

***Index Terms*—Learning from Demonstrations, Task and Motion Planning, Reactive Long-horizon Manipulation Planning**

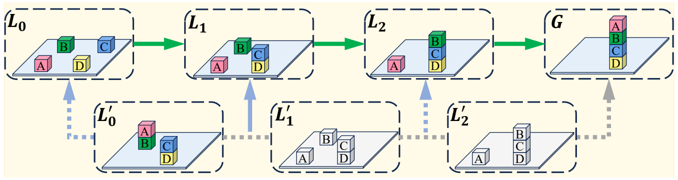

Fig. 1: Overview of Logic-LfD. Arrows refer to action primitives encoded with the proposed DMP variant (LQT-CP). The template task starts from $\mathcal{L}_0$ and ends at goal G. $\mathcal{L}_0 \to \mathcal{L}_1 \to \mathcal{L}_2 \to G$ illustrates the task-level demonstration. For a new task starting from $\mathcal{L}'_0$, a fixed sequential execution of actions primitives encoded by DMPs (green arrows) cannot transition from $\mathcal{L}'_0$ to the goal state G. Typical TAMP solvers find the action skeleton from $\mathcal{L}'_0$ to G from scratch (grey dashed long arrow). Instead, Logic-LfD tries to reach both task goal G and all other states (blue arrows) in the demonstration in parallel to find a feasible plan $\mathcal{L}'_0 \to \mathcal{L}_1$ connecting $\mathcal{L}'_0$ to the demonstration trajectory within the minimum time. It then merges the new plan with the corresponding segmentation of the demonstration $\mathcal{L}_1 \to \mathcal{L}_2 \to G$, thus accomplishing the new task faster than classical TAMP solvers.

I. Introduction

Learning from Demonstrations (LfD) has proven to be an effective approach for enabling robots to tackle complex manipulation tasks by imitating expert demonstrations [1]. Recent advancements in LfD have extended its applicability to long-horizon manipulation tasks, such as table rearrangement or kitchen activities, by segmenting demonstrations into sub-tasks [2], keyframes [3], or skills [4]. However, existing works often focus solely on reproducing demonstrations, assuming a sequential execution of learned skills can achieve task goals, which is not always applicable in real-world scenarios [5]. While traditional LfD methods like Dynamic Movement Primitives (DMP) [6], Task-parameterized Gaussian Mixture Model (TP-GMM) [7], or Dynamical Systems (DS) [8] can generalize demonstrated trajectories for new tasks or react to disturbances at the motion level, long-horizon planning necessitates not only motion-level generalization but also the ability to handle task-level variations and disturbances. Consequently, designing an LfD framework that can imitate, generalize, and react to disturbances while solving multi-step manipulation tasks remains a challenge [2, 5].

In the domain of long-horizon planning, Task and Motion Planning (TAMP) has emerged as a powerful tool for solving multi-step manipulation tasks [9, 10]. TAMP involves searching over a set of abstracted actions, with parameters often determined through sampling- or gradient-based motion planning. While theoretically capable of solving any feasible long-horizon manipulation task, TAMP relies on accurate dynamics modeling [11] and combinatorial searching of task instances and valid motions [2]. This limits the application of TAMP methods in real-world manipulation scenarios, involving model uncertainty and external disturbances.

In this work, we propose Logic-LfD, an LfD approach that generalizes one multi-step demonstration to solve new similar multi-step tasks, by combining the advantages of DMP and TAMP solvers. Logic-LfD leverages an optimal control formulation of DMP for motion modulation and extends it to incorporate via-point specifications for solving contact-rich manipulation sub-tasks, like pulling a cube with a hook in Figure. 3. Combining LfD with TAMP solvers can mitigate the challenges associated with modeling complex contact dynamics. Besides, we show that the demonstration can be formulated into multi-goal specifications for the TAMP solver so that it can try to achieve all the states on the demonstrated task-and-

Manuscript received: February 28, 2024; Revised May 9, 2024; Accepted June 17, 2024.

This paper was recommended for publication by Editor Kurniawati Hanna upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the State Secretariat for Education, Research and Innovation in Switzerland for participation in the European Commission's Horizon Europe Program through the INTELLIMAN project (https://intelliman-project.eu/, HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENSO project (http://sestosenso.eu/, HORIZON-CL4-Digital-Emerging Grant 101070310). We also acknowledge support from the China Scholarship Council (Grant No. 202106230104) and the SWITCH project (https://switch-project.github.io/), funded by the Swiss National Science Foundation.

The authors are with the Idiap Research Institute, CH-1920 Martigny, Switzerland and also with the EPFL, 1015 Lausanne, Switzerland; Authors with * contributed equally. E-mail:firstname.lastname@idiap.ch

Digital Object Identifier (DOI): see top of this page.

motion trajectory in parallel, which significantly accelerates the planning process while generalizing the demonstration for solving new tasks. We further extend our approach to introduce a reactive TAMP framework by deploying Logic-LfD in a closed-loop fashion for real-world tasks in dynamic environments.

Notably, our choice of DMP as the motion modulation block is motivated by its extrapolation generalization ability based on a single demonstration. However, conventional DMP suffers from poor via-point encoding ability, limiting its applicability for tasks with specific via-point requirements that can facilitate collision avoidance or contact-rich manipulation tasks [12, 13]. In our prior work [14], we propose Linear Quadratic Tracking with Control Primitives (LQT-CP), an optimal control formulation of DMP that can freely modulate how the system reacts to perturbations. In this paper, we explore the flexible motion modulation abilities of LQT-CP and further extend it to incorporate via-point tracking and generalization capability, addressing the limitations inherent in classical DMP.

In summary, our work proposes the following technical advancements:

- We propose Logic-LfD, a novel integration of TAMP solver with classical LfD methods. This integration extends DMP to solve long-horizon manipulation tasks by imitating multi-step demonstrations. It also explores the capability of LQT-CP to extend DMP to incorporate via-point tracking and generalization for addressing contact-rich manipulation sub-tasks;
- Logic-LfD improves the generalization ability of DMP to handle both motion- and task-level variants while solving similar multi-step manipulation tasks;
- We develop a Reactive TAMP approach, leveraging the fast generalization capability of Logic-LfD. This approach improves the reactivity of DMP to task-level disturbances, thus facilitating the solution of long-horizon manipulation tasks in dynamic environments.

II. Related Work

A. LfD for long-horizon manipulation tasks

Previous LfD methods typically address long-horizon manipulation tasks by segmenting demonstrations and composing movement primitives trained with the segmented demonstration [15, 16]. *Schwenkel et al.* and *Jaquier et al.* propose methods to find the best sequence of primitives based on demonstrations for the target tasks [17, 18]. Our Logic-LfD shares similarities in composing skill primitives for complex long-horizon manipulation tasks. However, unlike those methods, it does not assume a fixed sequential composition of those primitives (namely, without generalization or replanning) while solving the target long-horizon manipulation task in the real world. We show in Experiments (Section. V) with Tables I and II that the removal of this assumption can improve the generalization ability and reactivity of the system to handle task variants and disturbances, facilitating long-horizon manipulation planning in dynamic environments.

Under the umbrella of combining LfD with TAMP, *Mandlekar et al.* propose human-in-the-loop TAMP where human teleoperation assists TAMP solvers in finding feasible motions for complex contact-rich manipulation tasks without complicated dynamics modeling [11]. *Mcdonald et al.* and *Dalal et al.* [19, 20] propose to train a policy imitating a TAMP planner with a large-scale database of demonstrations collected offline, achieving good performance on generalization and reactivity. Moreover, large language models (LLMs) have proven beneficial for high-level task planning for long-horizon manipulation tasks [21, 22]. They can be considered as human language behavior models learned from internet-scale demonstrations. However, these language behavior models lack real-world experiences and robot information [23], necessitating great efforts on domain data collection and fine-tuning to find feasible trajectories for solving multi-step manipulation tasks with TAMP solvers in the real world [24]. In contrast, our Logic-LfD can handle various long-horizon manipulation tasks by imitating and generalizing a single demonstration.

B. Reactive Task and Motion Planning

Our closed-loop Logic-LfD operates within the domain of reactive TAMP, where rapid replanning is essential to respond to disturbances in dynamic environments. In recent studies [5] [25], temporal logic-based reactive synthesis has been combined with Dynamical Systems (DS) or behavior tree-based control strategies for reactive action selection and plan switching to address task-level disturbances in multi-step manipulation tasks. In contrast, our approach delves into a PDDL-based (Planning Domain Definition Language) TAMP planning framework, which includes logical state and action abstraction, thus facilitates the integration of action primitives learned from demonstration.

In [26], *Migimatsu et al.* propose to execute motions of the TAMP plan in object-centric Cartesian coordinates, demonstrating efficiency in reacting to motion-level disturbances. However, their method does not extend to handling task-level disturbances, setting it apart from our approach. In [27], *Xue et al.* introduce the Dynamic Logic-Geometric Program (D-LGP), showcasing impressive time efficiency for TAMP. However, D-LGP relies on keyframe-based action skeleton planning, making it incapable of handling disturbances within the keyframes. Receding-Horizon TAMP (RH-TAMP) approaches iteratively solve a reduced planning problem over a receding window of a shorter action skeleton, accelerating the TAMP planning process for handling interventions in changing environments [28, 29]. Nevertheless, a shorter prediction horizon increases the undesired infeasibility of action skeletons. Our Logic-LfD also accelerates TAMP by reducing the planning problem into a partial problem. However, it does not adopt a receding-horizon fashion, thus avoiding the influence on the feasibility.

In [30], *Harris et al.* propose a Feasibility-based Control Chain Coordination (FC^3) approach, showcasing impressive reactiveness by constructing offline a set of possible action plans and by switching between them in an online manner. However, the success of FC^3 heavily relies on the constructed plan library and switching strategy. The closest work to ours is Robust Logical-Dynamical Systems (RLDS) [31], which modifies the initial action plan by jumping backward or forward. However, RLDS assumes task disturbances can be handled without action plan switching, significantly limiting its reactiveness under substantial task perturbations. In several following experiments, our method demonstrates superior reactiveness to various levels of task disturbances than RLDS, which is achieved by an online generalization of the demonstrated nominal plan, without the need of constructing an action plan library offline as in [30].

III. Preliminary

A. Planning Domain Definition Language (PDDL)

In this section, we present essential concepts related to the Planning Domain Definition Language (PDDL) using the block stacking task illustrated in Figure 1 as an example. PDDL serves as an interface for defining the world by specifying a set of facts that describe relationships among objects in the environment. For instance, in $\mathcal{L}_1$ of Figure 1, the cube C is positioned on the cube D and this relationship is represented by the fact *(on C D)*. Throughout this paper, we use italicized symbols to denote these facts and the term *scene graph* to collectively represent the entire set of facts.

Among these facts, those that remain constant throughout the planning process is termed static facts (e.g., *(cube C)*). Conversely, facts that vary during the process are referred to as fluents (e.g., *(on C D)*, which changes with the robot's operation).

PDDL also provides the concept of action abstraction. An action is defined by a set of parameters, preconditions that must be satisfied before executing the action, and effects that occur after the action. Taking the action *pick* as an example, its parameters may include specifying which robot(s) pick which object(s). If, for instance, we have *(robot panda)* and *(cube C)* as the parameters, one precondition for executing this action should be *(clear C)*, indicating that there are no cubes on C, making it possible to grasp. The resultant effect is denoted by *(inhand panda C)*, signifying that cube C is now in the hand of the robot arm panda.

IV. Method

A. LQT-CP: an optimal control formulation of DMP

DMP is constructed in two parts to converge to the final goal position while tracking the acceleration profiles for mimicking the shape of a demonstrated trajectory. Similarly, Linear Quadratic Tracking (LQT), the most basic form of optimal control, is described by a cost typically composed of several parts acting at different state and control levels, with references either in the form of full trajectory or final goal positions. In particular, the cost can be specified to track a reference velocity or acceleration profile while reaching a target state at the end of the movement, with weight matrices balancing the importance of tracking the desired profiles and reaching the final goal position. With this similarity, we can reformulate DMP into a LQT fashion with the acceleration profile of the demonstrated trajectory as the reference trajectory and the attractor as the goal to be reached at the end.

Similarly to DMP, our system is a virtual point mass driven by a linear integrator system to describe the evolution of the system. As shown in [14], the control profile can be encoded with basis functions, resulting in a Control Primitive (CP) formulation of LQT that estimates superposition weights instead of the full list of control commands. The resulting LQT-CP yields a DMP by minimizing the cost function:

$$c = (\boldsymbol{\mu} - \boldsymbol{x})^\top \boldsymbol{Q} (\boldsymbol{\mu} - \boldsymbol{x}) + \boldsymbol{u}^\top \boldsymbol{R} \boldsymbol{u}, \tag{1}$$

where $\boldsymbol{\mu} = [\boldsymbol{\mu}_0^\top, \boldsymbol{\mu}_1^\top, \ldots, \boldsymbol{\mu}_T^\top]^\top$ indicates the concatenated vector of the position, velocity, and acceleration profiles of the demonstrated trajectory and $\boldsymbol{\mu}_T$ includes the goal position $\boldsymbol{g}$ to be reached at the end. $\boldsymbol{x} = [\boldsymbol{x}_0^\top, \boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_T^\top]^\top$ represents the concatenated system state variables. The matrices $\boldsymbol{Q} = \text{diag}(\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_T)$ and $\boldsymbol{R} = \text{diag}(\boldsymbol{R}_0, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_T)$ are a block-diagonal matrices showing the evolution of weight matrices $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ for variable $\boldsymbol{x}$ and command $\boldsymbol{u} = [\boldsymbol{u}_0^\top, \boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_T^\top]^\top$, correspondingly. The evolution of the state is described by a linear integrator $\boldsymbol{x}_{t+1} = \boldsymbol{A}_t \boldsymbol{x}_t + \boldsymbol{B}_t \boldsymbol{u}_t$, yielding $\boldsymbol{x} = \boldsymbol{S}_{\boldsymbol{x}} \boldsymbol{x}_0 + \boldsymbol{S}_{\boldsymbol{u}} \boldsymbol{u}$ at trajectory level, where $\boldsymbol{S}_x$ and $\boldsymbol{S}_u$ are composed of $\boldsymbol{A}_t$ and $\boldsymbol{B}_t$, see [14] for details.

Similarly as in DMP, where basis functions are used as movement primitives to represent the non-linear forcing term, basis functions can be used to represent the corresponding control commands. By leveraging a least squares formulation of recursive LQR, we showed in [14] that the control commands can be expressed as $\boldsymbol{u} = -\boldsymbol{F}\tilde{\boldsymbol{x}}_0 = -\boldsymbol{\Psi}\boldsymbol{W}\tilde{\boldsymbol{x}}_0$ where $\boldsymbol{\Psi}$ are the basis functions stored in matrix form, $\boldsymbol{W}$ is the weight matrix, and $\tilde{\boldsymbol{x}}_0$ is the augmented initial state for transferring the original LQT problem into an LQR problem with control primitives:

$$\begin{aligned} \min_{\boldsymbol{W}} &= \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{x}} + (\boldsymbol{\Psi}\boldsymbol{W}\tilde{\boldsymbol{x}}_0)^\top \boldsymbol{R} (\boldsymbol{\Psi}\boldsymbol{W}\tilde{\boldsymbol{x}}_0), \\ \text{s.t.} \quad & \tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{S}}_{\boldsymbol{x}} - \tilde{\boldsymbol{S}}_{\boldsymbol{u}} \boldsymbol{\Psi}\boldsymbol{W}) \tilde{\boldsymbol{x}}_0, \end{aligned} \tag{2}$$

where $\tilde{\boldsymbol{x}}$ is the concatenation of augmented state $\tilde{\boldsymbol{x}}_t = [\boldsymbol{x}_t^\top, 1]^\top$, $\tilde{\boldsymbol{Q}}$ is the concatenation of augmented weight matrices computed as

$$\tilde{\boldsymbol{Q}}_t = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{\mu}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_t & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{\mu}_t \\ \boldsymbol{0} & 1 \end{bmatrix},$$

$\tilde{\boldsymbol{S}}_x$ and $\tilde{\boldsymbol{S}}_u$ are state transfer matrices composed of $\tilde{\boldsymbol{A}}_t = \begin{bmatrix} \boldsymbol{A}_t & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix}$ and $\tilde{\boldsymbol{B}}_t = \begin{bmatrix} \boldsymbol{B}_t \\ 0 \end{bmatrix}$ that define the augmented system dynamics $\tilde{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{A}}_t \tilde{\boldsymbol{x}}_t + \tilde{\boldsymbol{B}}_t \boldsymbol{u}_t$.

Solving (2) results in the optimal weight matrix estimations:

$$\hat{\boldsymbol{w}} = (\boldsymbol{\Psi}^\top \tilde{\boldsymbol{S}}_{\boldsymbol{u}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{S}}_{\boldsymbol{u}} \boldsymbol{\Psi} + \boldsymbol{\Psi}^\top \boldsymbol{R} \boldsymbol{\Psi})^{-1} \boldsymbol{\Psi}^\top \tilde{\boldsymbol{S}}_{\boldsymbol{u}}^\top \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{S}}_{\boldsymbol{x}}, \tag{3}$$

The above solution can be used as a recursive formulation to generate a feedback controller that can handle external perturbation in real-time execution, providing a feedback controller $\boldsymbol{u}_t = -\tilde{\boldsymbol{K}}_t \tilde{\boldsymbol{x}}_t$ where $\tilde{\boldsymbol{K}}_t$ is the feedback gain matrix at time step t. We can therefore recursively calculate the feedback gains of our feedback controller LQT-CP with

$$\begin{aligned} \tilde{\boldsymbol{K}}_t &= \boldsymbol{\Psi}_t \hat{\boldsymbol{W}} \boldsymbol{P}_t, \\ \boldsymbol{P}_t &= \boldsymbol{P}_{t-1} (\tilde{\boldsymbol{A}}_{t-1} - \tilde{\boldsymbol{B}}_{t-1} \tilde{\boldsymbol{K}}_{t-1})^{-1}, \end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{K}}_0 = \boldsymbol{\Psi}_0 \boldsymbol{W}$, $\boldsymbol{P}_0 = \boldsymbol{I}$, see [14] for details.

LQT-CP provides additional flexibility in the design of the precision matrix $\boldsymbol{Q}$ to encode the importance of tracking different profiles of the demonstrated trajectory as well as capturing their correlation constraints. Therefore, we can design $\boldsymbol{Q}$ to track the acceleration profiles and goal position of the demonstrated trajectory to generate an optimal control formulation of DMP. With a single demonstration, the precision matrix Q can be arbitrarily high as long as the relative importance of those two terms is captured, similar to the specification of stiffness and damping gains in classical DMP. While multiple demonstrations are available, one can also set the precision $\boldsymbol{Q}$ as the inverse of the observed covariance so that the system reproduces the task with the same (co)variations as in the set of demonstrations [32, 33]. We further incorporate via-point specifications by simply indicating the via-points in the reference trajectory $\boldsymbol{\mu}$ and assigning the same importance to via-point and goal reaching in the weight matrix $\boldsymbol{Q}$. This simple extension allows us to address contact-rich manipulation subtasks while concurrently extending LQT-CP to handle long-horizon manipulation tasks. We illustrate this modification with the example of pulling a cube using a hook in Section V-C.

B. Logic-LfD

In this section, we introduce Logic-LfD for Task and Motion Planning (TAMP). The primary enhancement involves augmenting the classical demonstration representation, typically consisting of geometrical state sequences (e.g., position trajectories), with logical state and action abstractions derived from PDDL, and more importantly integrating TAMP solver with the proposed LQT-CP.

Consider a long-horizon manipulation task with N subtasks to be solved sequentially. The demonstration denoted as $\mathcal{P} = \{\{\mathcal{L}_i, \boldsymbol{a}_i\}_{i=1,...,N}, \{\boldsymbol{\tau}_k\}_{k=1,...,K}\}$, encompasses both task- and motion-level trajectories. Here, $\{\mathcal{L}_i\}_{i=1,...,N}$ represents the logical state sequence from the initial to the goal scene graph, akin to a task-level trajectory $\mathcal{L}$. $\boldsymbol{a}_i$ denotes the abstracted action, defining the transition from $\mathcal{L}_i$ to $\mathcal{L}_{i+1}$, and specifies which robot arm(s) executes the motion-level trajectory for manipulating which particular object(s). $\boldsymbol{\tau}_k$ represents the motion-level trajectory demonstrated for training the corresponding LQT-CP for the execution of the corresponding action primitive $\boldsymbol{a}_k$. In contrast to the straightforward propagation of geometrical states through Euclidean or unit quaternion operations ($\oplus, \ominus$), the logical state propagation is confined to a feasible action set $\mathcal{A} = \{a_k\}_{k=1,...,K}$ for the target task. Consequently, the corresponding demonstration for Logic-LfD should include not only the logical and geometrical state sequences but also the corresponding action sequence $\{\boldsymbol{a}_i\}_{i=1,...,N}$ indicating the feasible propagation of logical states.

Given the demonstration $\mathcal{P}$, we offline train the feedback gains of a set of LQT-CPs for each action primitive in $\mathcal{A}$ with the corresponding motion-level demonstrations $\{\boldsymbol{\tau}_k\}_{k=1,...,K}$. Replacing $\boldsymbol{\tau}_k$ in $\mathcal{P}$ with the corresponding LQT-CP$_k$, by exploring the generalization ability of LQT-CP at the motion level, we can generate an initial plan $\mathcal{P}_I$ which can handle new long-horizon manipulation tasks that only require the generalization of geometrical states. However, this limited generalization ability cannot address new tasks that require generalization in both task and motion levels.

Therefore, we further propose a new integration of TAMP solver with the proposed LQT-CP method. This integration results in Logic-LfD that quickly generalizes the demonstration $\mathcal{P}$ to solve new long-horizon tasks which share the same task goal state $\boldsymbol{G}$ but have different initial states than the template task, thus requiring generalization in both task and motion levels.

The overview of the whole framework is depicted in Figure 1, shown with a four-block stacking scenario. Supposing the new task starts with a new logical state $\mathcal{L}'_0$, Logic-LfD efficiently solves this new task with a two-step planning process and only needs to solve a partial TAMP problem. The first step involves finding a feasible task-and-motion trajectory $\mathcal{P}^{\text{new}}$ that transitions $\mathcal{L}'_0$ to its closest logical state $\mathcal{L}_1 \in \mathcal{L}$ in the demonstration $\mathcal{P}$, then concatenating $\mathcal{P}^{\text{new}}$ with the corresponding segmentation of the initial plan $\boldsymbol{\mathcal{L}_1} \rightarrow \boldsymbol{\mathcal{L}_2} \rightarrow \boldsymbol{G}$ to establish a feasible task plan transitions $\mathcal{L}'_0$ to $\boldsymbol{G}$ within the smallest time cost. Secondly, Logic-LfD generates the feasible motion-level trajectories by exploring the generalization ability of the proposed LQT-CP. $\mathcal{P}^{\text{new}}$, the state closest to the new initial state $\mathcal{L}'_0$ is indirectly determined by reformulating the demonstrated task-level trajectory $\mathcal{L} = \{\mathcal{L}_i\}_{i=1,...,N}$ as multiple goals $\mathcal{G}$ (referring to the *MultiGoalsSpecification* function in the Algorithm 1) and running the *PDDLStream* [9] algorithm to reach these goals concurrently, with the one reached within the minimum time considered as the target closest state. It is worth noting that our Logic-LfD framework treats the TAMP solver as a black-box tool, allowing for the integration of any TAMP solver into the framework, demonstrating its generality.

C. Reactive TAMP with Logic-LfD

Logic-LfD ensures fast discovery of a feasible task and motion plan. Since the closest state is usually reached much earlier than the goal state, Logic-LfD often resolves only a partial TAMP planning process while reacting to task-level disturbances. Considering that the planning time of a TAMP solver tends to increase significantly with the length of the final action skeleton, solving only a partial TAMP problem allows Logic-LfD to perform notably faster than the original TAMP solver. Therefore, we further extend the Logic-LfD into a close-loop TAMP framework (Algorithm 1) which promptly reacts to task- and motion-level disturbances while executing the nominal plan in dynamic environments.

It assumes the task goal G, demonstration $\mathcal{P}$, and an initial plan $\mathcal{P}_I$ are given. In the initialization process, multi-goal specifications $\mathcal{G}$ are generated based on the demonstration and the task goal with the designed *MultiGoalsSpecification* function in Line 4. This function filters out the static facts in $\mathcal{L}$ and keeps the sequence of fluents in $\mathcal{P}$. The designed *SceneGraph* function generates the facts about the current environment based on captured information about objects and robots in the environment and outputs the filtered key facts.

Algorithm 1: Reactive TAMP with Logic-LfD

```
1  Given: Environment $env$, Demonstration $\mathcal{P}$, Task Goal
     $\boldsymbol{G}$, Initial Plan $\mathcal{P}_I$
2  Output: next action $\boldsymbol{a}$ and motion $\boldsymbol{\tau}$
3  Initialization:
4  $\mathcal{G} = \mathit{MultiGoalsSpecification}(\mathcal{P}, \boldsymbol{G})$
5  $\mathcal{S}_C = \mathit{SceneGraph}(env)$
6  $\mathcal{P}_C \leftarrow \mathcal{P}_I$
7  while $\boldsymbol{G} \notin \mathcal{S}_C$ do
8      $flag, id = \mathit{LogicIn}(\mathcal{S}_C, \mathcal{G})$
9      if $flag$ is True then
10         $\boldsymbol{a} = \mathcal{P}_I[id]$
11     else
12         $\mathcal{P}^{\text{new}} = \mathit{PDDLStream}(\mathcal{S}_C, \mathcal{G})$
13         $\boldsymbol{a} = \mathcal{P}^{\text{new}}[0]$
14     $\boldsymbol{\tau} = LQT\_CP(\boldsymbol{a})$
15     $\mathcal{S}_C = \mathit{SceneGraph}(env)$
```

To solve the target task in a closed loop, we continuously estimate the current logical states $\mathcal{S}_C$ after executing each action $\boldsymbol{a}$ in the current plan $\mathcal{P}_C$. If the target goal $\boldsymbol{G}$ is not a subset of $\mathcal{S}_C$, indicating the task is not achieved yet. We then invoke the Logic-LfD to swiftly generate a new plan. With the *LogicIn* function in Line 8, we check if the current state $\mathcal{S}_C$ is a subset of the multi-goal specifications $\mathcal{G}$. If it is, we also generate its corresponding index *id* in the sequence. Then, the action $\mathcal{P}_I[id]$ in the initial plan will be a feasible action that transitions the current state $\mathcal{S}_C$ to the goal state $\boldsymbol{G}$. If $\mathcal{S}_C$ is not a subset of $\mathcal{G}$, it indicates a severe task-level disturbance in the environment, and *PDDLStream* with multi-goal specification $\mathcal{G}$ is applied to find a new feasible plan $\mathcal{P}^{\text{new}}$ that converges to the demonstration $\mathcal{P}$. The first action in $\mathcal{P}^{\text{new}}$ is executed as the feasible action for the current state. The action $\boldsymbol{a}$ not only shows which action should be executed, but also indicates which robot arm(s) should execute that action for manipulating which object(s). The LQT-CP is then applied in an object-centric manner to generalize the reference trajectory for generating the corresponding motion-level trajectory $\boldsymbol{\tau}^{\text{new}}$ to execute action $\boldsymbol{a}$.

V. Experiments

In this section, we present a comparative analysis between Logic-LfD and several baselines across three long-horizon manipulation planning problems. Figure 2 shows the experiment setups and the demonstrated tasks for the benchmarks. For each benchmark, we demonstrated the robot with an initial task and motion plan based on available action sets. Subsequently, we introduced distinct initial setups or disturbances to assess the performance of Logic-LfD against the selected baselines.

A. Benchmarks

*1) Tower Construction (**B1**):* This experiment involves controlling a robotic arm to manipulate a set of blocks using actions from the set $\mathcal{A} = \{pick, place, stack, unstack\}$ to arrange them in a specific order. Using the example of

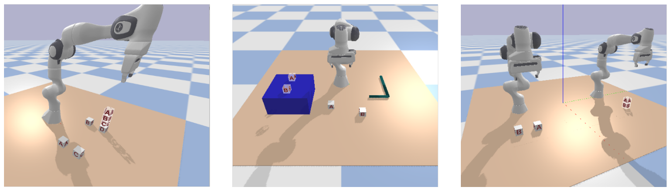
(a) B1 (b) B2 (c) B3

Fig. 2: Experimental setups for the three benchmarks. *B1*: Tower Construction, *B2*: Workspace Reach, *B3*: Dual-arm Box Transportation. Each sub-figure illustrates the demonstrated task, with the initial and task goal states depicted in grey and white color, respectively.

constructing a four-block tower, the template task begins with an initial state where all four blocks are placed on the table (i.e., without stacked blocks). The task goal state is achieved when the blocks are stacked in the sequence *A on B on C on D*. During the interaction between the robot and the blocks, both task- and motion-level constraints should be considered. For instance, the action *pick* for block *A* is only permissible if it is *clear* (no blocks on it) and if a *collision-free* trajectory exists.

*2) Workspace Reach (**B2**):* As an extension of the Tower Construction benchmark, this scenario involves the robot arm manipulating blocks with tools to address more complex long-horizon manipulation tasks. Blocks may be positioned beyond the reach of the robot arm. The template task begins with an initial state where a few blocks and one hook are placed on the table within the robot arm's workspace. It ends with the task goal state, where all blocks are placed on the shelf on the table. Actions *stack* and *unstack* are excluded to simplify the definition of the *pull* action which describes instances where the hook is grasped by the robot arm to extend its workspace and pull the block within its original workspace for picking or placing.

*3) Dual-arm Block Transportation (**B3**):* This benchmark further extends the Tower Construction experiment to validate the proposed algorithm in more complex multi-arm scenarios. The feasible action set $\mathcal{A} = \{pick, place, stack, unstack\}$ remains the same, but with augmented parameters introducing two *arms*. This complexity leads to more instantiations of abstracted actions and results in a more intricate long-horizon manipulation planning task. The template task begins with an initial state where all blocks are on the left table without any blocks on top and ends with the task goal state where blocks are expected to be stacked in a given sequence as in B1.

B. Disturbances

To comprehensively evaluate the generalization capability and reactivity of Logic-LfD across all benchmarks, we apply variants on initial states or disturbances at any states at different levels.

- L1 *Motion-level Variant/Disturbance:* Blocks are subjected to disturbances in different positions, while logical states remain consistent with the original or expected conditions;
- L2 *Slight Task-level Variant/Disturbance:* Blocks undergo disturbances, resulting in a different logical state. This

logical state aligns with that seen in the demonstration for the template task;

L3 *Severe Task-level Variant/Disturbance:* Blocks are disturbed to a novel and previously unseen logical state in the demonstration;

L4 *Extreme Task-level Variant/Disturbance:* A new block is introduced into the scene, thus significantly influencing the execution of the initial plan to achieve the target goal.

C. Comparison between LQT-CP and DMP

In this section, we demonstrate the flexible motion modulation capabilities of LQT-CP, addressing two crucial sub-tasks in B2. The experimental results are depicted in Figure 3. In the initial picking sub-task, the robot arm is tasked with picking up a hook potentially placed at three distinct positions. Both LQT-CP and DMP exhibit comparable performance, effectively generalizing the reference trajectory to new goal positions, as showcased in Figure 3-*Top*.

In the subsequent pulling sub-task, the robot arm is required to pull cube A using the hook to three different goal positions represented by red, green, and blue cubes in Figure 3-*Bottom*. This task demands not only the generalization of the reference pulling trajectory to new goal positions but also the precise control of the hook passing through two crucial via-points (the top and the left-down corners of cube A) for successful pulling. Consequently, it necessitates more flexible motion modulation concerning via-points tracking and generalization. Figure 3-*Bottom* illustrates the actual position of the hook when expected to be at the via-points. Notably, LQT-CP adeptly generalizes the reference trajectory and via-points, successfully pulling cube A to the goal positions, while standard DMP encounters challenges, which highlights the flexibility of the proposed LQT-CP formulation of DMP for motion modulation.

D. Generalization Ability of Logic-LfD

In this section, we assess the generalization ability of Logic-LfD in comparison to two baseline methods: *Linear* and *PDDLStream*. Here, *Linear* refers to the linear execution of a fixed sequence of DMPs without adapting the action sequence when addressing new tasks. For a comprehensive evaluation, each benchmark comprises tasks that share the same task goal but vary in their initial states from the template task. For B1, we solved a four-block tower construction problem. For B2 and B3, tasks were simplified by utilizing only two blocks. Moreover, LQT-CP serves as the motion modulation method for all baselines to ensure a fair comparison. We analyzed the overall success rate and computation time of the three methods in solving 100 tasks with random initial states for each benchmark across various levels (L1-L4). The summarized experimental results are presented in Table I.

In summary, both Logic-LfD and PDDLStream successfully solved all the validation tasks, while the linear execution of DMPs achieves success in only approximately 25% of the tasks, primarily those featuring motion-level variants in the initial states. The results show that Logic-LfD can enhance the generalization ability beyond linear execution. Moreover,

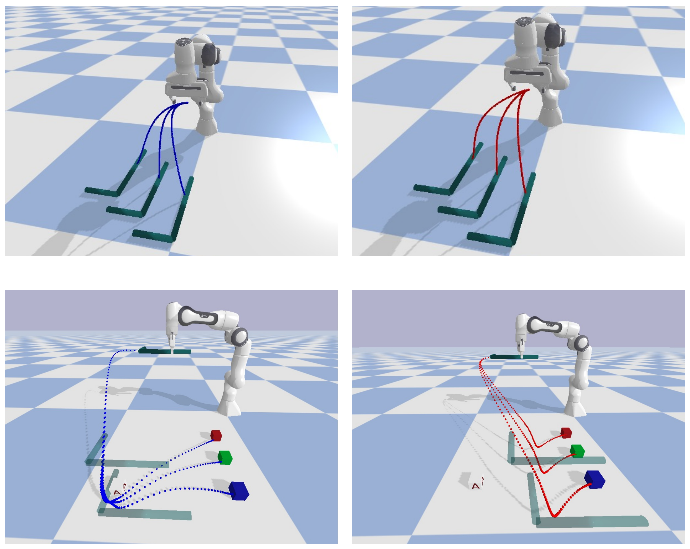

Fig. 3: Comparison between LQT-CP and standard DMP for two sub-tasks in the Workspace Reach benchmark (B2). *Top:* pick the hook. *Bottom:* pull cube A with the hook. The trajectories generated from LQT-CP are illustrated with blue lines in the left figures, and the ones for DMP are shown with red lines on the right. In the pulling task, the hook is expected to pass through two crucial via-points, the top and left-down corners of the cube A, for successfully accomplishing the task. In this figure, only the via-points for pulling cube A to the blue goal are shown with transparent hooks in the *Bottom* figures.

.

Logic-LfD exhibits faster planning capabilities than PDDLStream across all benchmarks, with a 30% to 40% improvement in B2 and B3, and a remarkable 70% improvement in B1. The substantial acceleration observed in B1 can be attributed to the inherently longer action sequence in the task, with which the planning time of PDDLStream increases significantly. Since Logic-LfD only needs to partially address the TAMP problem with a shorter action skeleton, it notably decreases the whole planning time, showing a significant acceleration in the experimental results. This reinforces the efficacy of Logic-LfD in handling long-horizon manipulation tasks and motivates us to formulate the reactive TAMP framework in Algorithm 1 by employing Logic-LfD in a closed-loop manner.

E. Reactivity of TAMP with Logic-LfD

In this section, we conduct a comparative analysis of the reactivity exhibited by the proposed closed-loop Logic-LfD framework and two baseline approaches: *Linear* and *RLDS*. The *Linear* baseline involves the linear execution of a predetermined sequence, without closed-loop feedback. *RLDS* is a reproduction of the algorithm outlined in [31], with the incorporation of the proposed LQT-CP for motion modulation. The experimental setup aligns with that detailed in Section V-D, ensuring consistency across benchmarks. We assess the performance of each method in handling disturbances at different levels (L1-L4) and report the average execution times based on 10 trials for each disturbed task in Table II.

Our analysis reveals that the proposed closed-loop Logic-LfD and RLDS demonstrate comparable reactivity in the

	Linear / Sequential			**PDDLStream**			**Logic-LfD**		
	B1	**B2**	**B3**	**B1**	**B2**	**B3**	**B1**	**B2**	**B3**
Success Rate	27%	31%	24%	100%	100%	100%	**100%**	**100%**	**100%**
Time [s]	N/A	N/A	N/A	0.53 ± 0.10	0.53 ± 0.23	0.23 ± 0.09	**0.16±0.07**	**0.37±0.27**	**0.14 ± 0.06**

TABLE I: Generalization ability comparison of Logic-LfD with baselines when solving tasks with various initial states in benchmarks. 'N/A' means no feasible values for the corresponding elements. Logic-LfD demonstrates superior generalization ability, achieving a higher success rate than the *Linear* execution of DMPs, and incurs significantly smaller computation time compared to PDDLStream.

	Linear / Sequential			**RLDS**			**Logic-LfD**		
	B1	**B2**	**B3**	**B1**	**B2**	**B3**	**B1**	**B2**	**B3**
L1	13.23±0.58	12.50±0.13	24.63±0.05	13.70±1.23	12.59±0.19	24.64±0.04	**13.23±0.35**	**12.50±0.13**	**24.62±0.03**
L2	N/A	N/A	N/A	18.07±0.95	19.08±0.01	30.83±0.04	**17.62±0.67**	**19.08±0.01**	**30.79±0.02**
L3	N/A	N/A	N/A	N/A	N/A	N/A	**23.05±0.49**	**26.03±0.78**	**30.23±0.08**
L4	N/A	N/A	N/A	N/A	N/A	N/A	**19.00±1.29**	**12.60±0.15**	**31.57±0.04**

TABLE II: Comparison of average execution time $[s]$ between Logic-LfD and baselines in three benchmarks. 'N/A' denotes no successful trials. Closed-loop Logic-LfD shows superior reactivity to various disturbances in all benchmarks. Please refer to the supplementary video through the project webpage for the illustration of simulation experiments for each element in the table.

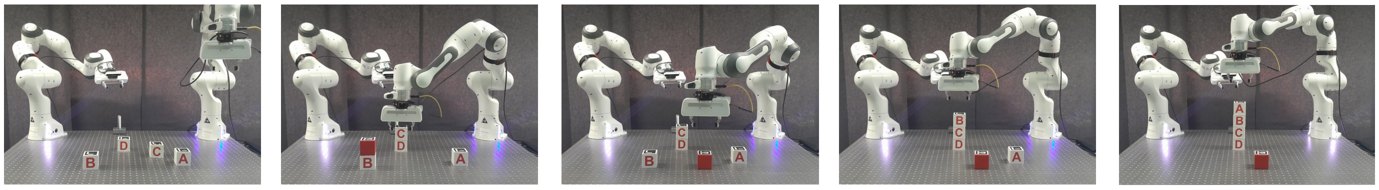

Fig. 4: Closed-loop Logic-LfD under extreme task-level disturbance (placing the red block on cube B after stacking cube C on D) in a real-world four-block stacking task. Logic-LfD reacts to the disturbance by unstacking the red block from cube B and placing it on the table, then continuing the original plan for achieving the goal state, as shown in the last figure.

presence of motion-level and slight task-level disturbances. Specifically, under L1 and L2 disturbances, both closed-loop Logic-LfD and RLDS effectively respond to the disturbances and achieve task goals within comparable execution times. However, RLDS relies on backward checking of the planned action sequence without plan modification, leading to challenges in handling disturbances at L3 and L4 across all benchmarks. In contrast, Logic-LfD exhibits the ability to generate a new feasible action plan when faced with those disturbances, enabling the TAMP approach to accomplish tasks even under severe or extreme disturbances. Notably, Logic-LfD proves to be faster in deriving the new action plan compared to the straightforward application of TAMP solvers, as demonstrated in the comparison results between PDDLStream and Logic-LfD in Section V-D, showing the significant reactivity inherent to the proposed closed-loop Logic-LfD approach.

F. Real-world Experiments

We assessed the reactivity of the proposed Logic-LfD in closed-loop scenarios for handling various task-level disturbances within the four-block Tower Construction scenario, as shown in Figure 4. All experiments were conducted using a 7-axis Franka Emika robot arm equipped with a RealSense D435 camera for object localization. The entire planning algorithm demonstrates the capability to respond to severe or extreme task-level disturbances at the frequency of 1Hz, providing corresponding actions and generalized motion sequences for rearranging the objects into the desired goal configuration. Additional demonstrations of the reactive behaviors under various task-level disturbances can be found in the supplementary video of the project webpage.

VI. Conclusion

In summary, we introduced Logic-LfD, an Learning from Demonstration (LfD) approach tailored for long-horizon manipulation tasks in dynamic environments. We leveraged an optimal control formulation of Dynamical Movement Primitive (DMP), Linear Quadratic Tracking with Control Primitives (LQT-CP), which naturally extends DMP to incorporate via-point specifications, proving beneficial for handling contact-rich manipulation sub-tasks. We demonstrated that Logic-LfD shows superior generalization ability than DMP and superior efficiency than TAMP solver in handling task variants during long-horizon planning. Consequently, we further extended this framework into a reactive TAMP system to quickly react to disturbances in dynamic environments. We validated the proposed methods through various simulation and real-world experiments, affirming the performances in term of skill transfer, generalization ability, and reactiveness to disturbances in long-horizon manipulation tasks.

In this paper, we integrated DMP with TAMP solver because of its exceptional extrapolation generalization ability with one single demonstration. Notably, this integration is not confined to DMP alone. It can be extended to other LfD algorithms. An interesting avenue involves further extending Logic-LfD by integrating with diffusion models [34] to tackle more complex long-horizon contact-rich manipulation tasks. Moreover, we aim to extend the proposed method to other real-world scenarios characterized by partially observable environments, which necessitate rapid replanning based on new observations.

One drawback of Logic-LfD is its potential to find slightly longer solutions when generalizing to new tasks compared to directly applying a TAMP solver. This is attributed to the consideration of goals in multi-goal specifications with the same priority. When multiple feasible solutions exist, Logic-LfD selects the one reached in the fastest way, but it doesn't guarantee that the chosen goal is the closest to the task goal state among all feasible solutions. Future work should explore strategies for assigning priorities to goals within the multi-goal specifications and the integration with optimization-based TAMP solvers [10, 27] to solve long-horizon manipulation tasks where the optimality of metrics (e.g. time and energy efficiency) is important.

References

[1] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Secaucus, NJ, USA: Springer, 2016, ch. 74, pp. 1995–2014, 2nd Edition.

[2] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," in *Proc. Robotics: Science and Systems (RSS)*, 2020.

[3] C. Pérez-D'Arpino and J. A. Shah, "C-Learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4058–4065.

[4] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[5] Y. Wang, N. Figueroa, S. Li, A. Shah, and J. Shah, "Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations," in *Conference on Robot Learning*. PMLR, 2023, pp. 94–105.

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

[7] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.

[8] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. Mit Press, 2022.

[9] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[10] M. Toussaint, "Logic-Geometric Programming: An optimization-based approach to combined task and motion planning." in *International Joint Conference on Artificial Intelligence*, 2015, pp. 1930–1936.

[11] A. Mandlekar, C. R. Garrett, D. Xu, and D. Fox, "Human-in-the-loop task and motion planning for imitation learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 3030–3060.

[12] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter-and extrapolation capabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4301–4308.

[13] S. Mghames, M. Hanheide, and A. Ghalamzan, "Interactive movement primitives: Planning to push occluding pieces for fruit picking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2616–2623.

[14] S. Calinon, "Programming industrial robots from few demonstrations." in *Human-Robot Collaboration: Unlocking the potential for industrial applications*. Institution of Engineering and Technology (IET), 2023, pp. 9–37.

[15] M. Saveriano, F. Franzel, and D. Lee, "Merging position and orientation motion primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7041–7047.

[16] S. Niekum, S. Chitta, A. G. Barto, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration." in *Robotics: Science and Systems*, vol. 9. Berlin, Germany, 2013, pp. 10–15 607.

[17] L. Schwenkel and M. Guo, "Optimizing sequences of probabilistic manipulation skills learned from demonstration."

[18] N. Jaquier, Y. Zhou, J. Starke, and T. Asfour, "Learning to sequence and blend robot skills via differentiable optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8431–8438, 2022.

[19] M. J. McDonald and D. Hadfield-Menell, "Guided imitation of task and motion planning," in *Conference on Robot Learning*. PMLR, 2022, pp. 630–640.

[20] M. Dalal, A. Mandlekar, C. R. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, "Imitating task and motion planning with visuomotor transformers," in *Conference on Robot Learning*. PMLR, 2023, pp. 2565–2593.

[21] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *arXiv preprint arXiv:2312.07843*, 2023.

[22] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[23] J. Xiang, T. Tao, Y. Gu, T. Shu, Z. Wang, Z. Yang, and Z. Hu, "Language models meet world models: Embodied experiences enhance language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[24] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter *et al.*, "PaLM-e: An embodied multimodal language model," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 8469–8488.

[25] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, p. 12618–12624.

[26] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, p. 844–851, 2020.

[27] T. Xue, A. Razmjoo, and S. Calinon, "D-LGP: Dynamic logic-geometric program for reactive task and motion planning," *arXiv preprint arXiv:2312.02731*, 2023.

[28] N. Castaman, E. Pagello, E. Menegatti, and A. Pretto, "Receding horizon task and motion planning in changing environments," *Robotics and Autonomous Systems*, vol. 145, p. 103863, 2021.

[29] C. V. Braun, J. Ortiz-Haro, M. Toussaint, and O. S. Oguz, "Rhh-lgp: Receding horizon and heuristics-based logic-geometric programming for task and motion planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 761–13 768.

[30] J. Harris, D. Driess, and M. Toussaint, "Fc 3: Feasibility-based control chain coordination," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 769–13 776.

[31] C. Paxton, N. Ratliff, C. Eppner, and D. Fox, "Representing robot task plans as robust logical-dynamical systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, p. 5588–5595.

[32] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, May-June 2014, pp. 3339–3344.

[33] Y. Zhang, F. Zhao, and Z. Liao, "Learning and generalizing variable impedance manipulation skills from human demonstrations," in *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2022, pp. 810–815.

[34] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," 2023.