Generalizable Long-Horizon Manipulations with Large Language Models

Haoyu Zhou[1], Mingyu Ding[2*], Weikun Peng[1], Masayoshi Tomizuka[2], Lin Shao[1*] and Chuang Gan[3,4]

***Abstract*— This work introduces a framework harnessing the capabilities of Large Language Models (LLMs) to generate primitive task conditions for generalizable long-horizon manipulations with novel objects and unseen tasks. These task conditions serve as guides for the generation and adjustment of Dynamic Movement Primitives (DMP) trajectories for long-horizon task execution. We further create a challenging robotic manipulation task suite based on Pybullet for long-horizon task evaluation. Extensive experiments in both simulated and real-world environments demonstrate the effectiveness of our framework on both familiar tasks involving new objects and novel but related tasks, highlighting the potential of LLMs in enhancing robotic system versatility and adaptability. Project website: `https://object814.github.io/Task-Condition-With-LLM/`.**

I. Introduction

Recent years have witnessed significant achievements in robot manipulations, and the growing demand for household and multifunctional robots capable of handling complex tasks has brought long-horizon manipulations into the spotlight.

Approaches like Task and Motion Planning [1–3] and hierarchical reinforcement/imitation learning methods [4, 5] propose to decompose long-horizon tasks into hierarchies, comprising high-level primitive tasks or discrete symbolic states alongside low-level manipulation motions. However, the full promise of long-horizon tasks necessitates not only task decomposition with primitive tasks but also a keen focus on environmental conditions. These conditions encompass aspects such as object interactions and spatial relationships, *e.g.*, an object inside another or a gripper grasping an object, playing an essential role in determining the success or failure of primitive tasks. They are crucial for guiding and correcting low-level trajectories and motions during long-horizon execution.

While such environmental conditions can be obtained through data sampling and point cloud or image processing from human demonstrations, two key limitations arise. Firstly, acquiring such data and demonstrations can be costly; and secondly, it is challenging to generalize to novel scenarios and tasks without additional demonstrations. Recently, Large Language Models (LLMs) have demonstrated strong reasoning abilities with human commonsense and in-context learning capabilities. For example, [6–9] showcase LLMs excel in failure explanation, task decomposition, and plan scoring for robot manipulation tasks.

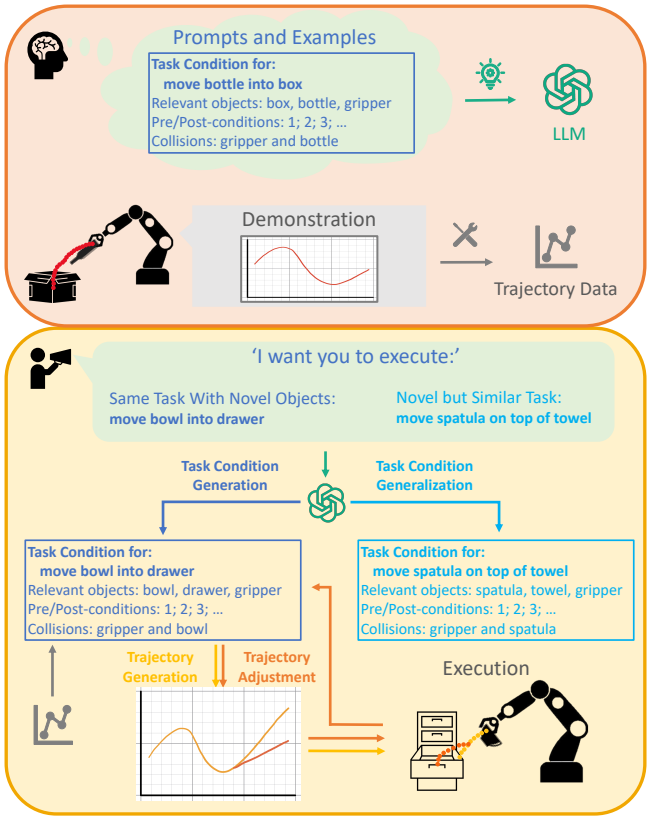

Fig. 1: Overall Framework. We leverage LLMs to generate and generalize primitive task conditions for both familiar tasks with novel objects and novel but related tasks. Subsequently, the high-level task conditions guide the generation and adjustment of low-level trajectories originally learned from demonstrations for long-horizon task execution.

Motivated by the above observations, this work leverages LLMs to create and generalize task conditions for both familiar tasks involving novel objects and new but related tasks. This offers promise in two ways: 1) Task conditions can be derived not only from human demonstrations but also from the inherent commonsense knowledge within LLMs, and 2) The prior knowledge and in-context learning capabilities of LLMs help generalize to novel objects and unseen tasks. Subsequently, the high-level task conditions guide the generation and adjustment of low-level Dynamic Movement Primitives (DMP) trajectories initially learned from demonstrations for long-horizon task execution. These components are seamlessly integrated and rigorously evaluated in our proposed manipulation task suite simulator and real-world environments. We summarize our **contributions** as follows:

[1]National University of Singapore `zhouhaoyu01@u.nus.edu, linshao@nus.edu.sg`
[2]UC Berkeley, USA `{myding, tomizuka}@berkeley.edu`
[3]University of Massachusetts Amherst, USA
[4]MIT-IBM Watson AI Lab, USA `ganchuang1990@gmail.com`
*Corresponding authors

- An LLM module for generating and generalizing task conditions for both seen and unseen primitive tasks.
- A systematic long-horizon manipulation framework that leverages high-level task conditions to guide the generation of low-level action trajectory based on DMPs.
- A challenging manipulation task suite within Pybullet.
- Extensive experiments in both simulated and real-world settings to demonstrate the effectiveness of our pipeline.

II. Related Work

A. Robotic Skill Learning for Long Horizon Tasks

A common approach to tackle long-horizon tasks is task-and-motion-planning (TAMP) that decomposes the planning process of a long-horizon task into discrete symbolic states and continuous motion generation [1–3]. However, classical TAMP methods rely on manually specified symbolic rules, thereby requiring known physical states with high dimensional search space in complex tasks. Recent works [10, 11] integrate learning into the TAMP framework to speed up the search of feasible plans [12–14] or directly predict the action sequences from an initial image [15].

Another potential solution for solving long-horizon tasks is hierarchical reinforcement/imitation learning [4, 5]. The options framework [16] and FUN (FeUdal Networks) [17], adopt modular architectures where agents learn both high- and low-level policies, with the high-level policy generating abstract sub-tasks and the low-level policy executing primitive actions. However, designing an efficient hierarchical structure and representing reusable domain knowledge across tasks remains a challenging problem. Relay Policy Learning [18] involves an imitation learning stage that produces goal-conditioned hierarchical policies, followed by a reinforcement learning phase that fine-tunes these policies to improve task performance. In this work, we explore using large language models as task condition generators for long-horizon robot manipulation, which are generalizable to both novel objects and novel tasks.

B. Large Language Models for Robotic Learning

LLMs, such as GPT-3 [19], PaLM [20, 21], Galactica [22], and LLaMA [23], exhibit strong capacities to understand natural language and solve complex tasks. For robotics, many research endeavors have focused on enabling robots and other agents to comprehend and follow natural language instructions [24, 25], often through the acquisition of language-conditioned policies [26–33]. Additionally, researchers also explores connecting LLMs to robot commands [7, 8, 33–38], leveraging pre-trained language embeddings [26, 28, 31, 39–45] and pre-trained vision-language models [46–48] in robotic imitation learning. The uniqueness of our work is to leverage GPT-3.5 to generate generalizable task conditions for long-horizon manipulation tasks.

III. Technical Approach

Our work presents a framework that leverages pretrained large language models (LLMs) to generate and generalize task conditions of primitive tasks, which are subsequently used to guide the generation and refinement of the Dynamic Movement Primitives (DMP) trajectories for long-horizon task execution. The framework is structured into four parts: **1) task condition generation** (Sec. III-B), **2) generalization** (Sec. III-C), which use LLMs to acquire task conditions for both seen and unseen tasks; **3) trajectory learning** (Sec. III-E) for generating DMP trajectories; and **4) trajectory generation & adjustment** (Sec. III-E), which incorporates information from task conditions and the environment.

```
Task name: move bottle into box
Relevant objects: bottle, box, gripper
Pre-conditions:
 gripper grasping bottle
 box open
Post-conditions:
 bottle inside box
 gripper inside box
 gripper grasping bottle
 box open
Collisions: gripper and bottle
```

Listing 1: Task condition structure.

A. Definitions

1) Task Condition Definition: In this work, the task condition for primitive tasks is focusing on the following aspects:

- **Task name:** a brief description of the primitive task.
- **Relevant objects:** potential objects relevant to the primitive task.
- **Pre-conditions:** environmental conditions that must be satisfied for the initiation of a primitive task, otherwise lead to task failure.
- **Post-conditions:** environmental conditions that mark the success of a primitive task.
- **Collisions:** permissible collisions during the task execution.

A typical example of task condition is shown in Lst. 1, the pre/post-conditions are divided into two categories:

- **Spatial relations:** shown in orange, representing spatial relations between relevant objects. The spatial relations we will focus on include: *inside, above, below, in front of*
- **Object states:** shown in magenta, representing the state of the relevant objects. The object states we will focus on include: *gripper: grasping/not grasping, and container: open/close*

2) Environment Information Acquiring: We enable the framework with a certain perception ability. The raw data we are getting from the environment is semantic point clouds generated from RGBD cameras. The following methods are then used to get different environmental information:

- **Object center position/bounding box:** we get the ground truth object center position and bounding box directly from Pybullet if it is visible in the semantic point cloud. A Gaussian noise is then added to each coordinate of the center position and bounding box, in order to simulate error in the real scenario.
- **Spatial relations:** the four spatial relations between objects mentioned in Sec. III-A.1 is generated based on center positions and bounding boxes of the objects, as in [6].
- **Object states:** the gripper state is judged through its spatial relations with other objects. If another is inside the gripper, we consider it to be grasped; the open or closed state of containers is judged by getting their moving joint position.
- **Collisions:** Collision between environment objects is detected using an implemented function in Pybullet. The position of the collision is generated at the same time.

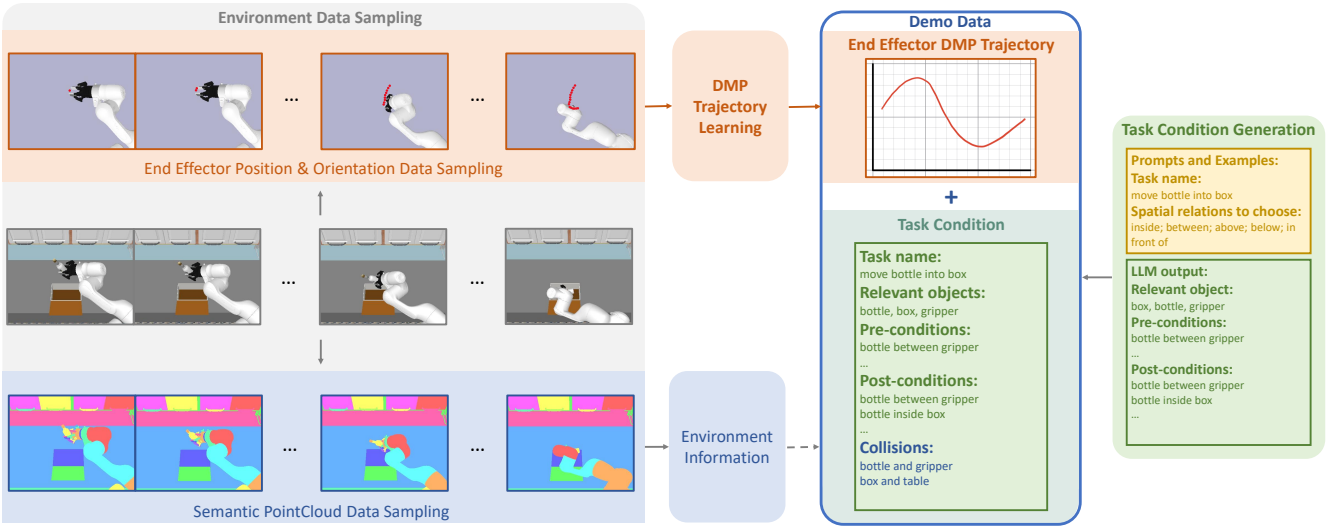

Fig. 2: Task Condition Generation and DMP Learning During Demonstration. This figure shows the outline of our framework during demonstrations. Task condition is generated by LLM with prompts and examples (green part), or from environment information (blue part) as comparison. The trajectory is encoded by DMP (orange part). Task condition and trajectory make up the demo data.

```
Imagine you are a spatial relation & collision judgment machine
Here are the spatial relations you can choose from:
  above, below, inside, in front of
Here are some examples:
[Task condition examples with only spatial relation & collision]
Remember to strictly follow the examples' format.
Q: Task name: [Generation Task Name]
```

Listing 2: Prompt for spatial relation & collision generation.

```
Imagine you are an object state judgment machine
Here are the object states you can choose from:
  gripper grasping/not grasping, container open/closed
Here are some examples:
[Examples of task condition with only object states]
Remember to strictly follow the examples' format.
Q: Task name: [Generation Task Name]
```

Listing 3: Prompt for object state generation.

3) Trajectory and Manipulator Control: In our framework, we sample and generate manipulator trajectories using end-effector pose in Cartesian coordinates and Euler angles. For manipulator control, we convert between Cartesian and joint space using Pybullet's built-in inverse kinematics. We employ position control, which, while straightforward, proves to be stable for the defined primitive tasks.

B. Task Condition Generation with LLMs

In order to generate task conditions of the same task with novel objects using LLM, we must overcome the well-known shortcoming of it being random and inconsistent. We managed to achieve this goal by first dividing the task condition generation problem into two parts: spatial relations & collisions generation problem, and object states generation problem. Two LLM chats are prompted differently to do these two parts separately. Also, the prompts we design emphasize the importance of remembering the examples given to it, and generating answers in the same format.

The prompt for spatial relations and collision generation is shown in Lst. 2. It has four components overall, including:

- an overall description of the job for LLM, indicated in black;
- spatial relations or object states to choose from, indicated in blue;
- examples in the form of Lst. 1 with only spatial relations included, indicated in orange;
- answer format indicated in pink.

```
Imagine you are a spatial relation & collision judgment machine
I will give you a task name describing a manipulator task, the
end effector of the manipulator is a gripper.
First, you should determine what are the relevant objects in
this task.
Then, you should present what spatial relations these objects
should have before (pre-conditions) and after (post-conditions)
the task is processed.
Here are the spatial relations you can choose from:
  above, below, inside, in front of
At last, you should present what collisions there might occur
during the completion of the task. You must only generate
collisions that include the relevant objects.
Here are some examples:
[Task condition examples with only spatial relation & collision]
Remember to strictly follow the examples' format
Do not generate any pre/post conditions except spatial relations
I gave you to choose from.
Q: Task name: [Generalization Task Name]
```

Listing 4: Chain-of-thoughts prompt for condition generalization.

Finally, we present a task name to LLM for task condition generation indicated by teal. To be mentioned, object names will be represented by letters (A, B, etc.) instead of specific object names. The prompt for object state generation remains consistent in structure and composition, except that changes in descriptions and examples are made for object state generation. Detailed prompt is shown in Lst. 3.

C. Task Condition Generalization with LLMs

Task condition generalization with LLM follows the prompting concept we mentioned in Sec. III-B, but should be able to generalize novel but similar task conditions. Therefore, changes in prompts are made in order to let the LLM loosen up a little in order to leverage its reasoning and learning ability from examples, meanwhile making sure the answer is in the same format as examples. We add the chain of thoughts prompting into the previous prompt in Lst. 2, telling the LLM how to reason step by step, indicated in purple in Lst. 4. The prompt for object states is similar and not presented here due to space constraints.

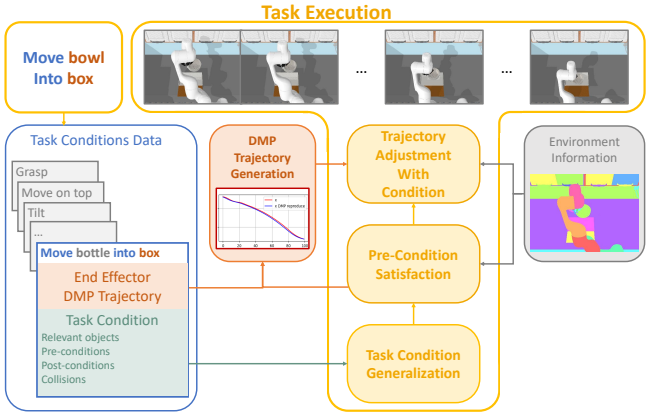

Fig. 3: Framework Flow Chat During Task Execution. This figure starts with the top-left primitive task we wish to execute, and the task condition of it is generalized based on generated task conditions. Further, information in the task condition and from the environment is used to help DMP generate and adjust end-effector trajectory, achieving execution of the given primitive task.

D. Trajectory Learning with DMP

Dynamic Movement Primitives (DMP) is a trajectory imitation learning method with high nonlinear characteristics and real-time performance ability. Meanwhile, it is capable of generating similar shaping trajectories to different goal positions. A large amount of work has been done based on the original formulation raised by [49, 50], and we choose to stick with the original discrete DMP, considering it is aimed to solve trajectory optimization problems in Cartesian space, and shows simplicity yet efficiency in our framework.

The basic formula of the discrete DMP is described by Eq. 1. y is the current system status (e.g. position) and $\dot{y}$, $\ddot{y}$ being its first and second derivatives. g is the goal status. The first term on the right is a PD controller with α_y and β_y representing the P parameter and D parameter and τ controlling the speed of convergence.

$$\tau^2 \ddot{y} = \alpha_y \left(\beta_y \left(g - y\right) - \tau \dot{y}\right) + f \tag{1}$$

The nonlinear term f is implemented by the normalized weighting of multiple nonlinear basis functions to control the process of convergence. The variable x in it satisfies a first-order system, making the nonlinear term time-independent. Eventually, it can be described by Eq. 2. N and w_i are the number and weight of the basis functions Ψ_i respectively, which we choose to be the Gaussian basis function described by Eq. 3, σ and c are its width and center position.

$$f(x,g) = \frac{\sum_{i=1}^{N} \Psi_i(x)\, w_i}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0), \quad \tau \dot{x} = -\alpha_x x \tag{2}$$

$$\Psi(x) = exp\left(\frac{-1}{2\sigma^2 (x-c)^2}\right) \tag{3}$$

In our work, we sample the trajectory of the end effector y, $\dot{y}$ and $\ddot{y}$ in each dimension mentioned in Sec. III-A.3, and the nonlinear term can be represented by rearranging Eq. 1 and put f to the left side. Then, with hyperparameters in Eq. 1 and Eq. 3 set as: $N = 100, \alpha_y = 60, \beta_y = \frac{\alpha_y}{4}, \alpha_x = 1, x_0 = 1$, we conducted the locally weighted regression method in [50] to calculate rest of the variables, and mathematical expressions for calculation will not presented here. The trajectory encoded by DMP will be part of the demo data shown in Fig. 2.

Algorithm 1 Pre-Condition Satisfaction

Require:
T_Cond - **T**ask Condition
function SATISFYPRECOND (*T_Cond*)
  currentConds ← FROMENVPC()
  preConds ← FROMCOND(*T_Cond*)
  for *preCond* **in** *preConds* **do**
    if *preCond* **not in** *currentCond* **then**
      newT_Name ← COND2TASK(*preCond*)
      newT_Cond ← GENCOND(*newT_Name*)
      SATISFYPRECOND (*newT_Cond*)
      CONTROLWITHCOND(*newT_Cond*)
    end if
    if *RUNTIME* > *60s* **then**
      break
    end if
  end for
end function

Algorithm 2 Trajectory Generation With Condition

Require: *T_Cond* - **T**ask Condition
function CONTROLWITHCOND(*T_Cond*)
  taskColl, postConds ← FromCond(*T_Cond*)
  targetPos ← FromEnvPC()
  goalPos ← *targetObjPos*
  trajDMP[:] ← GenDMP(*goalPos*)
  while *RUNTIME* ≤ *60s* **or** end(*trajDMP*) **or** postConds satisfied **do**
    ControlManipulator(*trajDMP*[+1])
    currentColl ← FromEnvPC()
    if *currentColl* not in *taskColl* **then**
      ControlManipulator(*trajDMP*[-10])
      collPos ← FromEnvPC()
      goalPos ← *goalPos* + (*goalPos* - *collPos*)
      trajDMP[:-1] ← GenDMP(*goalPos*)
    end if
  end while
end function

E. Trajectory Generation & Adjustment via Task Conditions

The method to generate trajectories using DMP is straightforward with parameters learned in Sec. III-D. With new starting position y_0 and goal position g given, we can use them to generate the nonlinear term described by Eq. 2 as well as position at any point by running a numerical simulation of the second-order system described by Eq. 1. In our framework, this method will be guided by corresponding task conditions to help generate and adjust trajectory. Two main functions are used sequentially, which are shown as yellow blocks in Fig. 3.

The first function is pre-condition satisfaction. It is conducted before reproducing a primitive task and is aimed to satisfy the pre-conditions of the task before execution. To achieve this, a recursive algorithm is used and its pseudo code is shown as Alg. 1. Given the task condition, the algorithm gets the pre-conditions in it and compares them to the current conditions in the environment acquired from environment information mentioned in Sec. III-A.2. Then, the pre-conditions that are not currently satisfied will raise a primitive task aiming to satisfy it. The mapping between

TABLE I: Task descriptions. Notations: PT – primitive task, LHT – long-horizon task, * – primitive task with novel objects.

PT1	Grasp Object	PT6	Fold Object
PT2	Release Object	PT7	Move Object A to position
PT3	Open Object	PT8	Move Object A On Top of Object B
PT4	Close Object	PT9	Move Object A into Object B
PT5	Tilt Object	PT10	Move Object A In Front of Object B
LHT1	Pick up a bottle, put the bottle into the box, close the box.		
LHT2	To put the bowl into the bottom drawer of the cabinet, first close the top drawer, then open the bottom drawer, put the bowl into the bottom drawer and then close the drawer.		
LHT3	Hold a mug up, place it on the table, and put toothpaste into the mug.		
LHT4	Grasp the spatula and place it onto the cloth, then fold the cloth.		
LHT1*	Pick up a bowl, put the bowl into the box, and close the box.		
LHT2*	To put the bowl into the upper drawer of the cabinet, first close the bottom drawer, then open the upper drawer, put the bowl into the upper drawer and then close the drawer.		

conditions and primitive tasks is done by humans in advance, for example, 'gripper grasping bottle' not satisfied will lead to 'grasp bottle' as a new primitive task, and its pre-condition will be generated and satisfied recursively.

The second function is trajectory adjustment with task conditions to prevent unintended collisions during execution. The pseudo-code is shown as Alg. 2. First, we generate a trajectory given the target object center from environment information as goal position using DMP, primitive tasks without a target object, such as moving or folding, goal position is defined by the difference between starting and ending positions during demonstration. During the execution of the trajectory, we monitor collisions in the environment. If there is an unwanted collision not in task condition, we stop executing and go backward 10 trajectory points. Then, we update the goal position regarding where the collision happens and adjust the rest of the trajectory. The algorithm terminates when all post-conditions are met or if the execution time exceeds a predefined threshold.

IV. Simulation

To evaluate our framework, we design a challenging Robotic Manipulation Task Suite in Pybullet [51]. The environment consists of two 7 Dof robots Franka and Kinova with a kitchen scene including various interactive objects. It contains 10 diverse primitive tasks (37 if considering different objects) and 4 long-horizon tasks in simulation.

These primitive tasks cover a broad spectrum of robotic manipulation skills. Our simulation includes 1) rigid object manipulation such as grasping and moving, 2) articulated object manipulation such as open/close the drawer/box, 3) periodic manipulation such as tilting the mug, 4) soft object manipulation such as grasp the cloth, and 5) dual-arm manipulation tasks, *e.g.*, folding the cloth. The detailed description of primitive tasks and long-horizon tasks is shown in Tab. I. We visualize our introduced simulator in Fig. 4 (left).

V. Experiments

A. Task Condition Generation & Generalization Experiment

First, we evaluate the ability of our framework to generate and generalize task conditions on all 10 primitive tasks. The LLM (GPT-3.5) is provided with condition examples.

TABLE II: Task condition generation and generalization.

PT Name	Generation		Generalization	
	LLM	FromEnv	LLM	FromEnv
Grasp	100%	100%	24%	-
Release	100%	100%	36%	-
Open	90%	**100%**	18%	-
Close	100%	100%	25%	-
Tilt	100%	100%	27%	-
Fold	**100%**	95%	30%	-
Move	81%	**100%**	40%	-
MoveInTo	90%	**91.7%**	32%	-
MoveOnTop	**90%**	86.7%	36%	-
MoveInFront	100%	100%	27%	-

TABLE III: Primitive task execution success rate.

PT Name	Baseline	w/o Cond	w/ Cond
Grasp	8.3%	87.5%	**93.1%**
Release	100%	100%	100%
Open	6.7%	**90%**	**90%**
Close	31.7%	**89.6%**	**89.6%**
Tilt	**86.7%**	75%	85%
Fold	0%	80.2%	**85%**
Move	16.65%	90.9%	**94.1%**
MoveInTo	68.8%	90.9%	**94.4%**
MoveOnTop	45.6%	81.9%	**91.1%**
MoveInFront	13.3%	**100%**	**100%**

Comparison is made with task conditions generated from environments. A successfully generated task condition should contain accurate and enough information to guide the execution of the primitive task. The result is shown in Tab. II.

When we evaluate the result of the experiment, the high success rate in task generation shows our properly prompted LLM has consistent abilities to accurately determine relevant objects and generate correct task conditions in an expected format. It even outperformed the success rate of generation using environmental information in some cases. When it comes to task condition generalization, the LLM still has a chance to figure out proper task condition for every primitive task, with the highest success rate being 40%. This shows our prompting method is feasible and gives the LLM task condition generalization ability. In contrast, another method clearly does not have the ability to generalize since no demonstration of the corresponding primitive task was given.

B. Primitive Task Execution Experiment

Secondly, we evaluate the ability of our framework to do imitation learning with DMP, and leverage task conditions to generate and adjust DMP trajectories. We run the execution of all 10 primitive tasks using three different methods:

- the image-based imitation learning baseline;
- with only DMP learning and generating trajectories;
- with correct task conditions generating and adjusting trajectories.

From Tab. III, the image-based baseline achieves much worse performance in all but PT5 execution, with ups and downs from 0 to 100, since it highly depends on good quality images with certain angles. In contrast, trajectories generated by DMP alone show better robustness with a consistent over 80% success rate in most primitive task execution. Then, when we compare methods [w/o task condition], we can see

TABLE IV: Evaluation of long-horizon tasks. '-' for infeasible.

Task Name	Baseline	w/o Cond	w/ Cond FromEnv	w/ Cond from LLM
Same as demonstration:				
LHT1	0.8%	50%	59.63%	**60.75%**
LHT2	0%	35%	**43.72%**	32.47%
LHT3	0%	20%	**31.80%**	21.25%
LHT4	0%	40%	45.30%	**49.50%**
Generalize to novel objects:				
LHT1*	-	-	-	**48.6%**
LHT2*	-	-	-	**28.8%**
Generalize to novel primitive tasks:				
LHT1	-	-	-	**19.5%**
LHT2	-	-	-	**10.1%**
LHT3	-	-	-	**7.7%**
LHT4	-	-	-	**15.5%**

the success rate on every primitive task is better if not the same when using our framework's method to generate and adjust trajectories. Though the improvement it makes seems not exceptional, it will make an impact when the success rate is to be multiplied during a long-horizon task.

C. Long-horizon Task Execution Experiment

Finally, we combine previous experiments to evaluate the overall ability of our framework to generate, generalize task conditions, then use it to generate and adjust DMP trajectory.

1) Experiment Setting: In order to improve the integrity of our execution experiment, and to maximize evaluation of different modules in our framework, we conduct the experiments under three cases. Before experiments, demo data including task condition and encoded trajectory of each primitive task is gained from demonstrations of long-horizon tasks (LHT1 to LHT4). And this experiment will take the success rate in Exp. V-A into consideration as well. The setting of the three cases for execution is as follows:

- *same as demonstration*: LHT1 to LHT4 using the same objects as in the demonstrations, with all example task conditions provided to the LLM.
- *generalize to novel objects*: LHT1* and LHT2* with novel objects, with all example task conditions from LHT1 and LHT2.
- *generalize to novel primitive tasks*: LHT1 to LHT4 with a randomly selected unseen (novel) primitive task, the example conditions excluding this primitive task are provided to LLM.

Similar to Exp. V-B, we run execution and evaluation under each case using four different methods:

- the image-based imitation learning baseline;
- with only DMP learning and generating trajectories;
- with task conditions from the environment to guide trajectories;
- with task conditions from LLM to guide trajectories.

2) Experiment Results: We first evaluate the result under *same as demonstration* case in Tab. IV. The performances of the baseline and with only DMP meet our expectations regarding Exp. V-B, since the success rate of a long-horizon task is approximately the product of that the primitive tasks have. The baseline performs even worse due to more inconsistent images during long-horizon tasks. As for methods [w/o task condition], the improvement in each primitive task adds up, leading to around 10% better with task condition

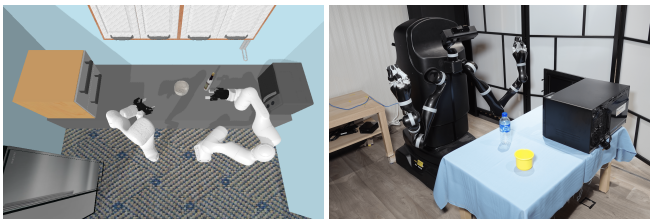

Fig. 4: Our simulator vs. real-world experimental setup.

generating and adjusting the trajectories, even considering the chance for failure in task condition generating. The difference between using environment information and LLM to generate task conditions depends on the result in Exp. V-A, with LLM still showing great competitiveness, outperforming in two long-horizon executions.

When it comes to other two cases, any slight backwardness of our framework before does not seem so important, as it is the only feasible method to do generalization to novel objects and primitive tasks. Only an average 8% decrease is noticed when generalizing the same task to novel objects. While for novel primitive tasks, because of the success rate shown in Exp. V-A experiencing a massive drop, there is an average decrease of 27.5%, leaving only around 13% success rate. But considering its difficulty, this is already a breakthrough compared to our other methods.

3) Advantage: We highlight an interesting advantage of our framework in executing long-horizon tasks. It has the ability to generate the needed trajectory on its own if an over far-sighted primitive task is given. For example, if we ask the manipulator to move a bottle before giving the instruction on grasping it, our framework can generate a grasping trajectory autonomously. Not until grasping is successfully finished will the manipulator begin moving it. This shows a certain intelligence our framework has and from certain aspects, can mean a higher success rate in long-horizon task execution.

D. Real-world Experiments

To demonstrate the practicality of our to perform long-horizon tasks in real scenarios, we set up an environment shown in Fig. 4 (right). It shows objects such as a bowl, a bottle that the manipulator can grasp, and a microwave that the manipulator can open and close. The environment features a dual-arm MOVO [52] robot which has two 7 DoF manipulators and a Kinect RGB-D camera overhead. We use Segment Anything [53] to obtain segmented point clouds of the surroundings and objects. Position control is performed as we use rangeIK [54] for solving inverse kinematics. Due to space constraints, we present our real-world experiments in the supplemental video and the project website.

VI. Conclusion

This work explores the potential of LLMs on primitive task condition generalization for generalizable long-horizon manipulations with novel objects and unseen tasks. The generated conditions are then utilized to steer the generation of low-level manipulation trajectories using DMP. A robotic manipulation task suite based on Pybullet is also introduced. We conduct experiments in both our simulation

and real-world scenarios, demonstrating the effectiveness of the proposed framework for long-horizon manipulation tasks and its ability to generalize to tasks involving novel objects or unseen scenarios. While our framework shows promise, there is room for improvement and a more versatile trajectory generator could complement our framework.

REFERENCES

[1] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1470–1477.

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

[3] M. Toussaint, "Logic-geometric programming: an optimization-based approach to combined task and motion planning," in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015, pp. 1930–1936.

[4] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete event dynamic systems*, vol. 13, no. 1, pp. 41–77, 2003.

[5] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in Neural Information Processing Systems*, pp. 3682–3690, 2016.

[6] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," *arXiv preprint arXiv:2306.15724*, 2023.

[7] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," *arXiv preprint arXiv:2209.11302*, 2022.

[8] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[9] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, *et al.*, "Grounded decoding: Guiding text generation with grounded models for robot control," *arXiv preprint arXiv:2303.00855*, 2023.

[10] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1gzR2VKDH

[11] K. Pertsch, O. Rybkin, F. Ebert, C. Finn, D. Jayaraman, and S. Levine, "Long-horizon visual planning with goal-conditioned hierarchical predictors," 2020.

[12] R. Chitnis, D. Hadfield-Menell, A. Gupta, S. Srivastava, E. Groshev, C. Lin, and P. Abbeel, "Guided search for task and motion plans using learned heuristics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 447–454.

[13] B. Kim, Z. Wang, L. P. Kaelbling, and T. Lozano-Pérez, "Learning to guide task and motion planning using score-space representation," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 793–812, 2019.

[14] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Active model learning and diverse action sampling for task and motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4107–4114.

[15] D. Driess, J.-S. Ha, and M. Toussaint, "Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image," in *Robotics: Science and Systems 2020 (RSS 2020)*. RSS Foundation, 2020.

[16] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[17] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3540–3549.

[18] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1025–1037.

[19] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[20] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," *CoRR*, vol. abs/2204.02311, 2022.

[21] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[22] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," *CoRR*, vol. abs/2211.09085, 2022.

[23] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *CoRR*, 2023.

[24] J. Luketina, N. Nardelli, G. Farquhar, J. N. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, "A survey of reinforcement learning informed by natural language," in *IJCAI*, 2019.

[25] M. Ding, Y. Xu, Z. Chen, D. D. Cox, P. Luo, J. B. Tenenbaum, and C. Gan, "Embodied concept learner: Self-supervised learning of concepts and mapping through instruction following," in *Conference on Robot Learning*. PMLR, 2023, pp. 1743–1754.

[26] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, "Concept2Robot: Learning manipulation concepts from instructions and human demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[27] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor, "Language-conditioned imitation learning for

robot manipulation tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 139–13 150, 2020.

[28] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn, *et al.*, "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation," in *Conference on Robot Learning*. PMLR, 2022, pp. 1303–1315.

[29] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, 2022.

[30] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning over unstructured data," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 205–11 212, 2022.

[31] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "BC-Z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning (CoRL)*, 2021, pp. 991–1002.

[32] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," *Conference on Robot Learning (CoRL)*, 2022.

[33] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "RT-1: Robotics transformer for real-world control at scale," *Robotics: Science and Systems (RSS)*, 2023.

[34] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *Conference on Robot Learning (CoRL)*, 2022.

[35] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," *arXiv preprint arXiv:2209.07753*, 2022.

[36] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "ChatGPT for robotics: Design principles and model abilities," *Microsoft Auton. Syst. Robot. Res*, vol. 2, p. 20, 2023.

[37] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023.

[38] A. Bucker, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, and R. Bonatti, "Latte: Language trajectory transformer," *arXiv preprint arXiv:2208.02918*, 2022.

[39] F. Hill, S. Mokra, N. Wong, and T. Harley, "Human instruction-following with deep reinforcement learning via transfer-learning from text," *arXiv preprint arXiv:2005.09382*, 2020.

[40] C. Lynch and P. Sermanet, "Grounding language in play," *Robotics: Science and Systems (RSS)*, 2021.

[41] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "VIMA: General robot manipulation with multimodal prompts," *International Conference on Machine Learning (ICML)*, 2023.

[42] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[43] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 894–906.

[44] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," *arXiv preprint arXiv:2203.12601*, 2022.

[45] Y. Mu, S. Yao, M. Ding, P. Luo, and C. Gan, "Ec2: Emergent communication for embodied control," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6704–6714.

[46] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, B. Zitkovich, F. Xia, C. Finn, *et al.*, "Open-world object manipulation using pre-trained vision-language models," *arXiv preprint arXiv:2303.00905*, 2023.

[47] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo, "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *arXiv preprint arXiv:2305.15021*, 2023.

[48] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[49] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[50] 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.

[51] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[52] Kinova, "Kinova-movo." [Online]. Available: https://docs.kinovarobotics.com/kinova-movo/Concepts/c_movo_hardware_overview.html

[53] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[54] Y. Wang, P. Praveena, D. Rakita, and M. Gleicher, "Rangedik: An optimization-based robot motion generation method for ranged-goal tasks," *arXiv preprint arXiv:2302.13935*, 2023.