LLM^3: Large Language Model-based Task and Motion Planning with Motion Failure Reasoning

Shu Wang[1⋆], Muzhi Han[1⋆], Ziyuan Jiao[2⋆†], Zeyu Zhang[2], Ying Nian Wu[1], Song-Chun Zhu[2], Hangxin Liu[2†]

***Abstract*— Conventional Task and Motion Planning (TAMP) approaches rely on manually designed interfaces connecting symbolic task planning with continuous motion generation. These domain-specific and labor-intensive modules are limited in addressing emerging tasks in real-world settings. Here, we present LLM^3, a novel Large Language Model (LLM)-based TAMP framework featuring a domain-independent interface. Specifically, we leverage the powerful reasoning and planning capabilities of pre-trained LLMs to propose symbolic action sequences and select continuous action parameters for motion planning. Crucially, LLM^3 incorporates motion planning feedback through prompting, allowing the LLM to iteratively refine its proposals by reasoning about motion failure. Consequently, LLM^3 interfaces between task planning and motion planning, alleviating the intricate design process of handling domain-specific messages between them. Through a series of simulations in a box-packing domain, we quantitatively demonstrate the effectiveness of LLM^3 in solving TAMP problems and the efficiency in selecting action parameters. Ablation studies underscore the significant contribution of motion failure reasoning to the success of LLM^3. Furthermore, we conduct qualitative experiments on a physical manipulator, demonstrating the practical applicability of our approach in real-world settings.**

I. Introduction

Sequential manipulation planning is an essential capability for robots to autonomously perform diverse tasks in complex environments. Executable motions must be effectively generated for robots to achieve long-term task objectives, requiring efficient planning algorithms for responsive operation and reasoning capabilities to anticipate environmental changes. Task and Motion Planning (TAMP) formulates a methodology that hierarchically decomposes planning into two stages: the high-level symbolic task planning stage reasons over long-horizon abstract action sequences, and the low-level continuous motion planning stage computes feasible trajectories subject to geometric constraints. In recent years, TAMP has enabled significant advances [1–7]. However, a persistent challenge remains to properly interface between the task planner and the motion planner to efficiently solve TAMP, *i.e.*, generating action sequences that satisfy both symbolic task goals and continuous motion constraints.

Traditional TAMP approaches often rely on manually designed modules to interface between symbolic and continuous domains, as depicted in Fig. 1(a). These modules serve two key roles. First, they act as action parameter samplers that generate real-valued parameters for symbolic actions. These parameters provide numerical goals to the motion planner. Selecting appropriate action parameters, *e.g.*, in the object rearrangement task, a suitable 2D target location (x, y) for action `Place(object)`, is crucial for the motion feasibility of actions and the efficiency of TAMP. While previous works propose to learn heuristic parameter samplers from data [8, 9], they are tailored to specific domains and lack generalizability. Second, these modules implement mechanisms to incorporate motion failure into the task planner to generate improved action plans, *e.g.*, by updating the symbolic state [10]. However, they usually require domain-specific design by human experts. In summary, these modules are domain-specific and require substantial manual effort to design, which hinders generalizability to novel environments.

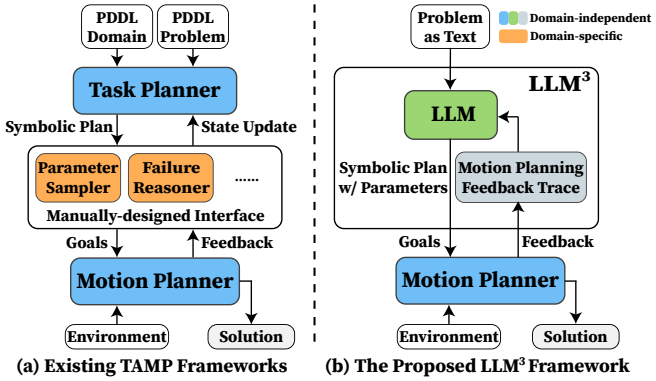

Fig. 1: **The proposed LLM^3 framework.** (a) Traditional TAMP frameworks rely on manually designed, domain-specific modules for interfacing between task and motion planners. (b) In contrast, we leverage a pre-trained LLM to iteratively propose refined plans and action parameters, by reasoning on motion planning failures.

Recent Large Language Models (LLMs) pre-trained on web-scale text data have demonstrated emergent capabilities in reasoning [11] and planning [12]. Pre-trained LLMs can perform task planning [13], generate continuous parameters [14], and reason on environment feedback [13]. Our intuition is that pre-trained LLMs could provide a general and domain-independent approach to interfacing between symbolic and continuous domains for TAMP, eliminating the need to design domain-specific modules manually.

In this paper, we present **LLM^3** (**L**arge **L**anguage **M**odel-based Task and **M**otion Planning with **M**otion Failure Reasoning), an LLM-powered TAMP framework that reasons over motion planning feedback for effective planning-Fig. 1(b). Specifically, LLM^3 employs a pre-trained LLM to (i) propose symbolic action sequences towards the task goal, (ii) generate continuous action parameters that lead to feasible motion, and (iii) reason over motion planning

This work was supported in part by the National Natural Science Foundation of China (No.52305007, 62376031).

⋆ Shu Wang, Muzhi Han, and Ziyuan Jiao contributed equally to this work. † Corresponding authors. [1] University of California, Los Angeles. [2] State Key Laboratory of General Artificial Intelligence, BIGAI.

feedback to iteratively refine the proposed symbolic actions and parameters. This framework offers several key advantages over traditional TAMP approaches. First, it does not require manually designed symbolic domain files for task planning, instead leveraging the knowledge encoded in the LLM to propose symbolic actions. Second, it uses the LLM as a domain-independent informed parameter sampler to generate continuous action parameters, which benefits from the implicit heuristics of the LLM [14]. Third, its reasoning over motion planning feedback is independent of the specific choice of planner. Crucially, we categorize and organize the possible motion planning feedback to feature two major motion failure modes, *i.e.*, collision and unreachability. Such motion planning feedback allows LLM^3 to refine the generated action sequences and parameters in a more targeted way, and find a feasible TAMP solution with fewer planning iterations and motion planning queries.

We evaluate LLM^3 in a simulated tabletop box-packing task, which poses challenges in reasoning about potential failure modes, collisions, and unreachable areas, throughout the sequential manipulation planning problem. Quantitative results demonstrate the effectiveness of LLM^3, with ablation studies verifying: (i) reasoning over motion feedback significantly improves success rates and planning efficiency, and (ii) the LLM-based parameter sampler is substantially more sample efficient than a random sampler. Furthermore, we conduct real-robot experiments to show that LLM^3 can be applied to real-world problems.

In summary, our contributions are threefold:

1) We introduce LLM^3, the ***first*** TAMP framework that holistically employs a pre-trained LLM as a domain-independent task planner, informed action parameter sampler, and motion failure reasoner.
2) We categorize and organize the feedback from the motion planner, which enables LLM^3 to efficiently identify and resolve planner-independent motion failures through targeted refinement of actions and parameters.
3) We conduct comprehensive experiments in both the simulation and the real world to demonstrate the effectiveness of LLM^3 in solving TAMP problems.

A. Related Work

Task and Motion Planning: Conventional TAMP approaches employ a high-level task planner to generate symbolic action sequences and a low-level motion planner to generate motion trajectories. The task planner requires predesigned symbolic planning domains represented in formatted representations, such as Planning Domain Definition Language (PDDL). Significant efforts have been made to develop manually engineered modules that interface the task planner and motion planner, such as incorporating motion-level constraints into task planning [5, 15], making approximations at the motion level [16, 17], and designing specialized communication modules [10]. However, manually defining task planning domains and interface modules to fully capture real-world complexity is impractical. Furthermore, as the action space grows, searching for geometrically feasible symbolic action sequences becomes computationally challenging without effective heuristics [15]. Recent work has explored data-driven heuristics to improve TAMP efficiency [8, 18], but these domain-specific heuristics lack generalizability across domains. In this work, we employ a pre-trained LLM as both the task planner and the interface between task and motion. We expect that semantic knowledge in the LLM can provide domain-independent heuristics for TAMP.

Robot Planning with LLMs: Recent Large Language Models (LLMs) [19, 20] encode vast world knowledge and exibit the emergent capability for planning [12]. Pre-trained LLMs have been applied for task planning of robots or embodied agents [13, 21–27]. Notably, Inner Monologue [13] takes in textualized environment feedback and generate actions to execute, while ReAct [28] further advanced this closed-loop approach by integrating reasoning and acting. Voyager [22] and DEPS [21] focus on developing open-ended embodied agents that iteratively replan based on execution failure. Furthermore, LLMs have been applied to solve TAMP problems [29, 30]. Specifically, LLM-GROP [29] employs a pre-trained LLM to instantiate symbolic goals and continuous object placements for semantic object rearrangement, which are then fed into a classical TAMP planner. AutoTAMP [30] leverages an LLM to translate natural-language task specifications into a formal language processable by off-the-shelf TAMP algorithms. Our usage of LLMs in TAMP is inspired by many of the above works; however, the major difference is that we leverage the LLM as the core component of our TAMP framework.

II. Preliminaries and Problem Setting

A. Task and Motion Planning

A *TAMP* problem is a tuple $< \mathcal{O}, \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0, g >$, where:

- $\mathcal{O}$ is the set of objects in the environment.
- $\mathcal{S}$ is the state space factorized with respect to objects $\mathcal{O}$, where a state $s \in \mathcal{S}$ comprises the state of all objects, *e.g.*, their 3D positions and dimensions.
- $\mathcal{A}$ is the set of primitive actions with low-level execution handled by a motion planner. A primitive action $a \in A$ is parameterized by object variables $\bar{o}$ and continuous parameters θ, which can be instantiated with specific objects $\underline{o} = (o_1, ..., o_m)$ and parameter values to produce a ground action $\underline{a}$, *e.g.*, `Place(block,`$[0.1, 0.2]$`)`. The parameters θ provide the goal for the motion planner; we say $\underline{a}$ is feasible when the motion planner has a solution, *i.e.*, a collision-free trajectory τ. We denote a feasible action by $\underline{a}(\tau)$.
- $\mathcal{T}$ is the state transition function that outputs the next state s_{t+1} after executing an action $\underline{a}_t(\tau_t)$ at a state s_t, *i.e.*, $s_{t+1} = \mathcal{T}(s_t, \underline{a}_t(\tau_t))$. We assume that $\mathcal{T}(s_t, \underline{a}_t(\tau_t)))$ can be evaluated with a black-box simulator.
- s_0 is the initial state that follows $s_0 \in \mathcal{S}$.
- g is the goal function $g\colon\ \mathcal{S} \to \{0, 1\}$, which checks whether the task goal is achieved at state s.

The objective of TAMP is to derive a sequence of feasible actions $(\underline{a}_0(\tau_0), \underline{a}_1(\tau_2), ..., \underline{a}_T(\tau_T))$, such that $g(s_{T+1}) = 1$ and $s_{t+1} = \mathcal{T}(s_t, \underline{a}_t(\tau_t)))$, where $t = 0, 1, ..., T$.

A common strategy to solve the TAMP problem is known as "search-then-sample" [1, 3, 10], which alternates generating symbolic action sequences through backtracking search

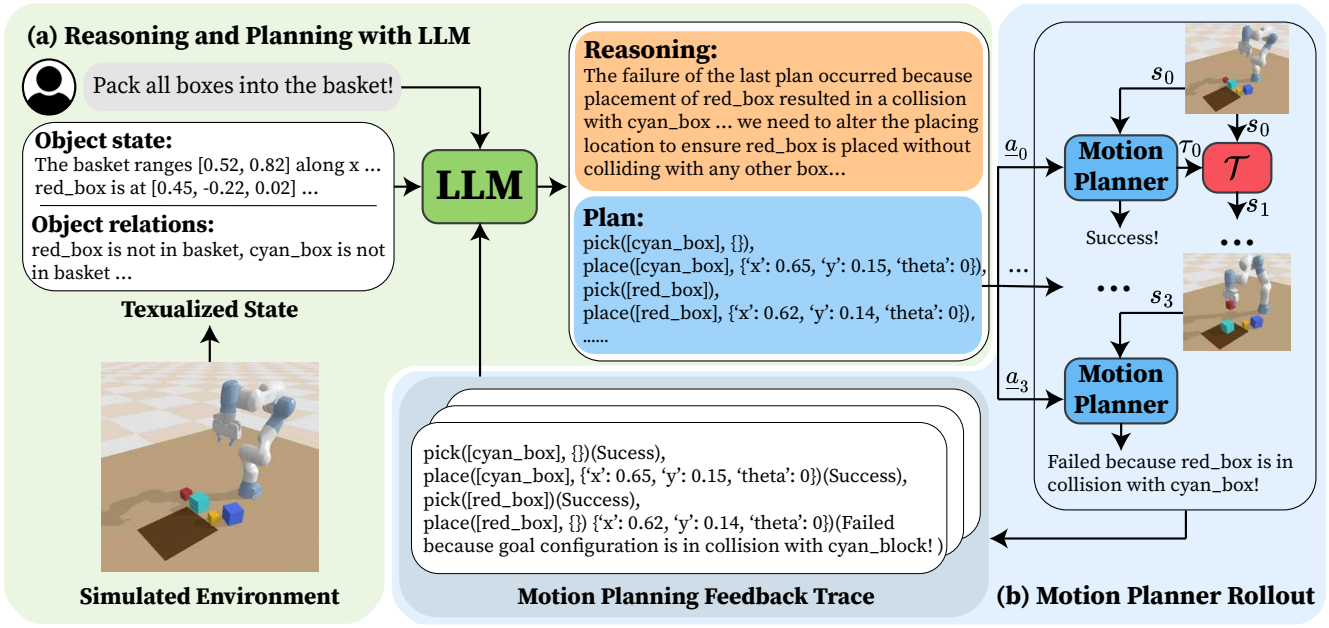

Fig. 2: **System diagram of the proposed LLM^3 framework.** (a) We show an example of utilizing a pre-trained LLM for reasoning and generating action sequences. (b) The feasibility of the proposed action sequence is verified by rollout with a motion planner and transition function $\mathcal{T}$. The motion planning feedback is saved into a trace that is provided to the LLM in the next iteration.

and sampling the continuous action parameters until a feasible plan is found to reach the goal. However, a naive search-then-sample task and motion planner is usually inefficient, as verifying the action feasibility requires invoking the computationally expensive motion planning process [31, 32]. To mitigate this complexity, researchers have crafted symbolic domains [10, 33, 34] to prune infeasible symbolic action sequences, or learned heuristic samplers to sample action parameters [9, 35] that lead to feasible plans. In this work, we leverage pre-trained LLMs as both a task planner and a heuristic sampler to generate symbolic action sequences and action parameters.

B. Planning as Sequence Prediction

Recently, planning problems have been formulated as sequence prediction to avoid the computationally expensive search process [36]. In particular, pre-trained LLMs are employed to generate discrete actions [12] and continuous parameters in an auto-regressive manner when provided in-context prompts. Formally, with textualized initial state s_0, goal g and additional context c, we have:

$$\begin{aligned} \underline{a}_{0:T} &= \arg\max \; p_{LM}(\underline{a}_{0:T} | s_0, g, c) \\ &= \arg\max \; p_{LM}(\underline{a}_0 |, s_0, g, c) \prod_{t=1}^{T} p_{LM}(\underline{a}_t | \underline{a}_{0:t-1}, s_0, g, c), \end{aligned} \tag{1}$$

where p_{LM} is the generative probability of a pre-trained language model. Following this scheme, we use pre-trained LLMs to propose sequences of symbolic actions and continuous action parameters based on the initial state, goal, and trace of motion planning feedback. As the proposed action sequences may be infeasible at the motion level, we validate the feasibility of proposed action sequences with a motion planner, the state transition function $\mathcal{T}$ and the goal g. We iterate these two steps until a feasible plan is found.

III. Method

We introduce LLM^3, a TAMP framework that leverages a pre-trained LLM to reason on motion failure and generate iteratively refined symbolic actions and continuous action parameters. The system diagram of LLM^3 is shown in Fig. 2. Below, we elaborate on the overall framework, reasoning and planning with the pre-trained LLM, and the designed motion planning feedback.

A. The LLM^3 Framework

As shown in Algorithm Algorithm 1 and Figure 2, the LLM^3 framework iterates between: (i) reasoning on previous motion failure and generating an action sequence (*i.e.*, symbolic actions and continuous parameters) with a pre-trained LLM, and (ii) verifying the feasibility of the action sequence with a motion planner. At each planning iteration, the LLM takes the current state s and the trace of motion planning feedback $trace$, and outputs the reasoning for the previous motion failure $reason$ and an action sequence llm_plan to solve the TAMP problem (see Section III-B). The motion planner then attempts to find a collision-free motion trajectory for each action $\underline{a} \in llm\_plan$ sequentially. We synthesize motion planning feedback (see Section III-C) for each motion planner query and aggregated the feedback for all actions in llm_plan. The planning iteration ends with failure when an action has no feasible motion or the action sequence fails to reach the goal. The aggregated feedback is then added to a trace $trace$, which is maintained throughout the life cycle of the framework with a maximum size k. In the next planning iteration, the trace is fed into the LLM to generate motion failure reasoning and another action sequence that improves on the previous one. This process repeats until a generated action sequence reaches the goal with no motion failure or the maximum number of attempts is exceeded.

Algorithm 1: LLM^3 for TAMP

Input : pre-trained LLM with prompt template to generate action sequences LLM, state transition function $\mathcal{T}$ with motion planner MP, goal function g, initial state s_0, maximum number of planning attempts N_{max}, maximum trace size k

Output: success indicator $success$, a sequence of feasible actions $plan$

// initialize output and feedback trace

$plan \leftarrow []$, $success \leftarrow$ False, $feedback \leftarrow []$, $trace \leftarrow []$

// initialize current state and planning iteration count

$s \leftarrow s_0$, $iter \leftarrow 0$

// main planning loop

while *not* $success$ *and* $iter < N_{max}$ **do**

  // reason and plan with LLM

  $reason, llm\_plan \leftarrow LLM(s_0, trace)$, $iter \leftarrow iter + 1$

  // rollout llm_plan from s with motion planner

  foreach $\underline{a} \in llm\_plan$ **do**

    // call motion planner to verify feasibility of $\underline{a}$

    $\tau, mp\_feedback \leftarrow MP(s; \underline{a})$

    $feedback.append((\underline{a}, mp\_feedback))$

    // terminate rollout if action $\underline{a}$ is infeasible

    if τ *is* $None$ **then**

      break

    // update state s and add $\underline{a}$ to $plan$

    $s \leftarrow \mathcal{T}(s, \underline{a}(\tau))$, $plan.append(\underline{a}(\tau))$

  // check whether s satisfies goal

  $success, task\_feedback \leftarrow g(s)$

  // when the plan is feasible but doesn't reach goal

  if *not* $success$ **then**

    // record $task\_feedback$

    $feedback.append(task\_feedback)$

    // add aggregated feedback to $trace$

    $trace.append(feedback)$, $trace \leftarrow trace[-k:]$

    // clear aggregated feedback

    $feedback \leftarrow []$

return $success$, $plan$

Overall, the LLM^3 framework can be regarded as a search-then-sample TAMP planner that generates action sequences with incrementally improved quality, guided by the intrinsic heuristics of the pre-trained LLM and the previous motion failure. By design, we expect LLM^3 to exhibit superior efficiency compared to unguided planners that sample action parameters randomly.

B. Reasoning and Planning with pre-trained LLM

Following previous attempts in utilizing LLMs for reasoning and planning [11–13], we prompt a pre-trained LLM to generate motion failure reasoning and action sequences in text format. Since we want to limit the domain-specific prior provided to the LLM, we use zero-shot prompting without providing any planning examples. We adopt Chain-of-Thought (CoT) prompting to have the LLM generate the reasoning and action sequence in an auto-regressive manner. The prompt fed into LLM comprises the following contents: (i) a system message that provides the global context to the pre-trained LLM and activates its planning capability, (ii) a task description specifying the environment, goal, and available primitive actions, (iii) the textualized initial environment state, (iv) the trace of motion planning feedback, and (v) the output format. Note that most promp content will remain unchanged for different planning iterations of the same TAMP problem. Only the motion planning feedback trace in (iv) will be updated as new failed action sequences are added. Fig. 2(a) illustrated an example of reasoning and planning with LLM.

We implement two strategies for the pre-trained LLM to generate a new action sequence that improves on the previous one: (i) *backtrack*, where we expect the LLM to backtrack to a previous action that has feasible motion, and continual to generate actions that complete the plan, and (ii) *from scratch*, where we expect the LLM to directly generate a new action sequence that attempts to avoid the motion failure happened to its previous output. We achieve this by designing the system message and output format description in the prompt. We show the prompt template for the two variants in Fig. 3, where the content specific to each variant is highlighted.

System message: You are an AI robot that generates a plan of actions to reach the goal... You are expected to correct the plan incrementally (on top of the last plan) to avoid motion failure. This may involve sample new parameters for the failed action or reverse one or more succeeded actions for backtracking... You are expected to generate a plan from scratch.

Task description: A robot arm is tasked to pack boxes into a basket on a table. The robot sits at (0, 0), and faces the positive x-axis, while the positive z-axis points up. The robot is equipped with primitive actions, each taking a list of objects and continuous parameters as input:

- `pick`([`obj`], {}): pick up `obj`, with no parameters.
- `place`([`obj`], {"x": $[0.0, 1.0]$, "y": $[-1.0, 1.0]$, "$theta$": $[-3.14, 3.14]$}): place `obj` at location (x, y) with the planar rotation theta, where x ranges $(0.0, 1.0)$, y ranges $(-1.0, 1.0)$, and $theta$ ranges $(-3.14, 3.14)$.

Initial state: {init_state}

motion planning feedback trace: {trace}

Output format: Please generate output step-by-step, which includes your reasoning for the failure of the last plan as well as the generated plan... Answer the questions: (i) what is the cause of the failure of the last plan? (ii) can altering action parameters for the failed action solve the problem... (iii) do we need to reverse one or more succeeded actions executed before the failed action... (ii) what is your strategy to generate a new plan from scratch to accomplish the task goal? Please organize the output following the JSON format below: { "Reasoning": "My reasoning for the failure of the last plan is ...", "Full Plan": ["pick(['red_box'], {})", "place(['red_box'], {'x': 0.51, 'y': 0.02, 'theta': 0.00})", ...] }

Fig. 3: **Prompt templetes used by LLM^3.** We show alternative contents specific for the *backtrack* variant in orange and *from scratch* variant in blue.

C. Synthesizing Motion Planning Feedback

We realize a ground action $\underline{a}$ by calculating a collision-free trajectory τ with a sampling-based motion planner, *e.g.* Bi-directional Rapidly-exploring Random Trees (BiRRT) [37]. The input to the motion planner includes the initial environment state (includes the robot state), and a goal pose of the robot end effector specified by the continuous action parameters of $\underline{a}$. The motion planner samples and searches for a collision-free joint space trajectory for the robot to reach the goal end-effector pose.

By default, the motion planner reports a binary signal that indicates whether there is a feasible trajectory. It does not give more abstract-level feedback, which explains why motion planning fails. As a result, the TAMP planners can acquire useful feedback from motion planning failures to improve high-level planning. To this end, we additionally synthesize semantically meaningful motion-level feedback so that LLM^3 can improve on previous failures more effectively. We observe that typical motion planning failures can be categorized into two types, *i.e.*, collisions and unreachability. Therefore, we synthesize categorized motion planning feedback following the templates below (Fig. 4):

(A) The goal configuration is in collision with `object`.
(B) The goal configuration has no feasible IK solution.
(C) The goal configuration is collision-free and reachable.

In practice, we integrate the motion planner with an additional IK solver and collision checker for obtaining these feedbacks, finding this design to be practically effective.

IV. Simulation and Experiment

In simulations, we initially perform an ablation study on our LLM^3 framework in two settings of the tabletop box-packing task, quantitatively evaluating its effectiveness based on i) the planning success rate (%SR), ii) the number of LLM calls (#LM), iii) and the number of motion planner calls (#MP). Additionally, we demonstrate the role of LLM as an informed action parameter sampler by comparing it to a baseline utilizing random sampling strategies. Finally, we validate the proposed LLM^3 framework through experimentation on a perception-integrated physical robotic manipulator, confirming its validity in real-world scenarios.

A. Simulation Setup

The simulation has an important role in Robotics [38, 39]. We developed a PyBullet-based simulation environment for our box-packing tasks, as illustrated in Fig. 5a. In **Setting 1**, three different sets of objects are given with increasing total sizes, while the basket size remains unchanged. This task requires the LLM^3 to oversee potential collisions among objects and the robot throughout the action sequence. The LLM must reason why collisions occur and adjust previous actions to ensure feasible task and motion plans. **Setting 2** involves placing the Set 3 objects into baskets of increasing

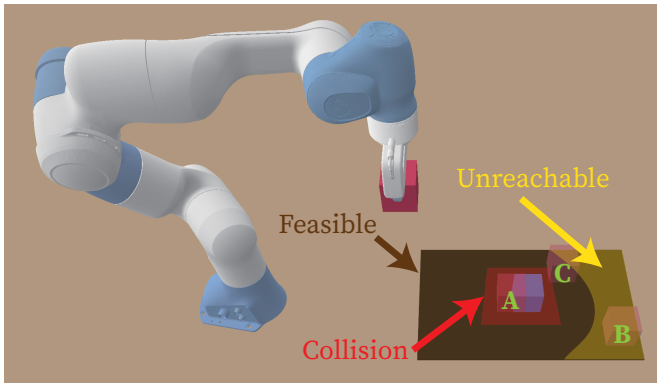

Fig. 4: **Three types of motion possibilities.** A: the object placement is in collision with an existing object. B: the object placement is beyond the robot's reach. C: the object placement is feasible.

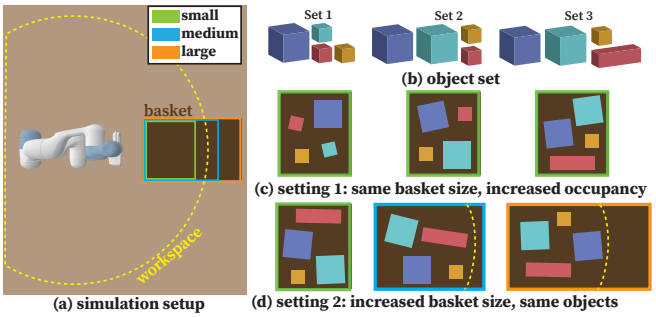

Fig. 5: **The box-packing task setup in a simulated environment.** (a) The task requires the robot to place one of (b) three sets of objects fully into the basket. (c) In setting 1, the total object size increases but the basket sizes remain the same. All baskets are fully reachable by the robot. (d) In setting 2, the basket size increases, but some portions of baskets are longer within the robot's reach.

sizes. Here, the robot cannot access the entire basket region but encounters a collision likelihood similar to the most crowded condition in Setting 1. In this setting, the task becomes more complex as it challenges the LLM to reason about the robot's operational space and adjust the plan accordingly. Throughout the simulations, we utilize GPT-4 Turbo as the LLM planner and BiRRT [37] as the motion planner, with 10 attempts for each setting. [1]

B. Ablation Study

The conducted ablation study compares the proposed LLM^3 with baseline methods:

1) **LLM^3 Backtrack**: The proposed LLM^3 framework *backtrack* variant. See Fig. 3
2) **Backtrack**: The LLM proposes plans with backtracking but without motion planning feedback (line 7).
3) **LLM^3 Scratch**: The proposed LLM^3 framework *from scratch* variant. It replans the entire action sequence, incorporating feedback from the motion planner if any action fails. See Fig. 3
4) **Scratch**: The LLM plans the action sequence once and executes the plan without any feedback.

Three evaluation criteria are considered: The number of LLM calls (#LM) counts how many times the LLM API is called during planning. A lower #LM indicates that the planner can produce a feasible task plan more efficiently. In each attempt, #LM has a maximum cap of 20; attempts with over 20 #LMs will be counted as a failure. The total success rate is recorded as %SR. Additionally, the number of motion planner calls, #MP, is another critical criterion as in traditional TAMP approaches, where massive and time-consuming motion planner calls are the main cause for their inefficiency. The study results are summarized in Table I.

In both settings, integrating motion planning feedback results in a decrease in #LM and #MP, along with an increase in %SR for both *backtrack* and *scratch* strategies. This indicates that the LLM can reason about failures from motion planning feedback, and importantly, propose adjusted task plans and action parameters that are more likely to produce feasible motions. Surprisingly, no clear evidence suggests that utilizing backtracking is superior to replanning from

[1]The code is available at https://github.com/AssassinWS/LLM-TAMP.

TABLE I: Ablation Study

Method	Setting 1									Setting 2								
	Easy			Medium			Hard			Small			Medium			Large		
	%SR	#LM	#MP	%SR	#LM	#MP	%SR	#LM	#MP	%SR	#LM	#MP	%SR	#LM	#MP	%SR	#LM	#MP
LLM^3 Backtrack	**100**	**1.6**	**11.8**	**100**	**4.4**	**28.4**	60	11.4	39.8	60	11.4	39.8	**80**	**9.5**	**50**	50	13.5	44.8
Backtrack	100	1.8	12.6	90	6.3	32	40	15.1	55.3	40	15.1	55.3	30	14.6	16	30	15.8	48
LLM^3 Scratch	100	1.7	13.2	100	7	46.1	**70**	**8.8**	**30.9**	**70**	**8.8**	**30.9**	70	11.5	50.2	**60**	**10.6**	**32**
Scratch	100	2.4	17.6	60	12.3	45.3	50	13.7	42.8	50	13.7	42.8	30	16.2	45.7	40	13.2	24

scratch. We've observed distinct behaviors among different strategies employed by LLM^3. Specifically, when utilizing backtracking, LLM sometimes consistently adjusts the action parameter of the specific action that *directly* led to failure, without adjusting previous actions. In contrast, the LLM^3 employing a planning from scratch strategy simply re-samples all actions and parameters, occasionally resulting in a slightly lower #LM.

C. Action Parameter Selection

This study examines whether an LLM can function as an informed action parameter sampler, using the **Medium** setup from **Setting 1** in the previous section. In this study, the action sequence is predetermined and consists of a total of 8 steps, where the robot sequentially performs pick and place actions to relocate all four blocks into the basket. However, the specific placement location is not provided, requiring the action parameters to be sampled to ensure the feasibility at the motion level. We implement three methods, each runs 50 attempts, for comparison:

1) **Random Samples**: A random sampler is implemented to independently and uniformly sample all the block placement locations within the basket region.
2) **LLM**: The LLM is provided with the task setting and the symbolic action sequence, prompting it to select the action parameters to make the action sequence feasible.
3) **LLM + Feedback**: Building upon 2), the LLM is additionally provided with motion planning feedback if the previous attempt fails.

The results summarized in Table II demonstrate the efficacy of using LLM as an informed action parameter sampler in the context of box-packing tasks. Specifically, while random sampling requires an average of 109.6 iterations and 663.1 #MP to achieve feasible action sequences, the LLM substantially reduces them to an average of 10.8 iterations and 70.2 #MP. When incorporating motion planning feedback alongside the LLM, the sampling requirement decreases even further to an average of 7.9 iterations and 53.2 #MP. These findings underscore the ability of LLMs to efficiently select action parameters that align with task constraints, resulting in a notable reduction in the number of motion planning calls needed to generate feasible action sequences. Moreover, the additional benefit gained from integrating motion planning feedback highlights the importance of incorporating real-time feedback mechanisms to further refine and optimize the sampling process. Overall, these results highlight the potential of LLMs as a valuable tool to improve the efficiency and effectiveness of robotic manipulation tasks, particularly in scenarios where efficient action parameter selection is crucial for improving planning performance.

TABLE II: Comparison of Different Sampling Strategies

Method	#Itration	#MP
Random Samples	109.6	663.1
LLM	10.8	70.2
LLM + Feedback	**7.9**	**53.2**

D. Experiment Setup

To validate the effectiveness of our proposed method in a real-world setting, we conducted an experiment using a Franka Research 3 manipulator. The goal was to demonstrate the robot's ability to perceive and manipulate objects in an environment with uncertainties in both perception and execution. The robot observed a single point cloud from a third-person-view RGB-D camera, capturing the workspace containing various objects such as blocks and a plate. To identify and locate individual objects, we employed Grounded Segment Anything [40] for object segmentation, initially segmenting objects in the 2D RGB image and then projecting the results onto the corresponding 3D point cloud. This approach yielded per-object point clouds, essential for planning and executing manipulation tasks.

Fig. 6 presents a qualitative evaluation of our method, where the robot was tasked with placing all blocks on the plate. The results demonstrate that our method enabled the robot to successfully identify and manipulate objects despite the uncertainties and challenges in the cluttered environment. The successful execution of this experiment validates the practicality and robustness of our approach, showcasing its potential for various real-world applications such as object sorting, assembly tasks, and household assistance.

V. Conclusions

In this paper, we introduced LLM^3, a new TAMP framework powered by the pre-trained LLM. LLM^3 leverages the rich knowledge encoded in and the powerful reasoning capability processed by LLMs to (i) propose action sequences based without requiring a prior planning domain, (ii) generate continuous action parameters for the robot motion planner, and more importantly, (iii) refine task and/or motion plans in response to motion planning failures.

Following validation through various simulations and experiments, we demonstrated that LLM^3 effectively produced and refined task and motion plans for box-packing problems, exhibiting promising potential in addressing previously unspecified tasks. Our study also revealed that although the pre-trained LLM can generate action parameters more efficiently than random samplers, it still necessitated multiple feedback iterations and motion planner calls. Looking ahead, the incorporation of in-context learning or fine-tuning techniques holds promise for further enhancing its efficiency, representing a crucial step towards empowering robots to tackle emerging tasks in real-world scenarios.

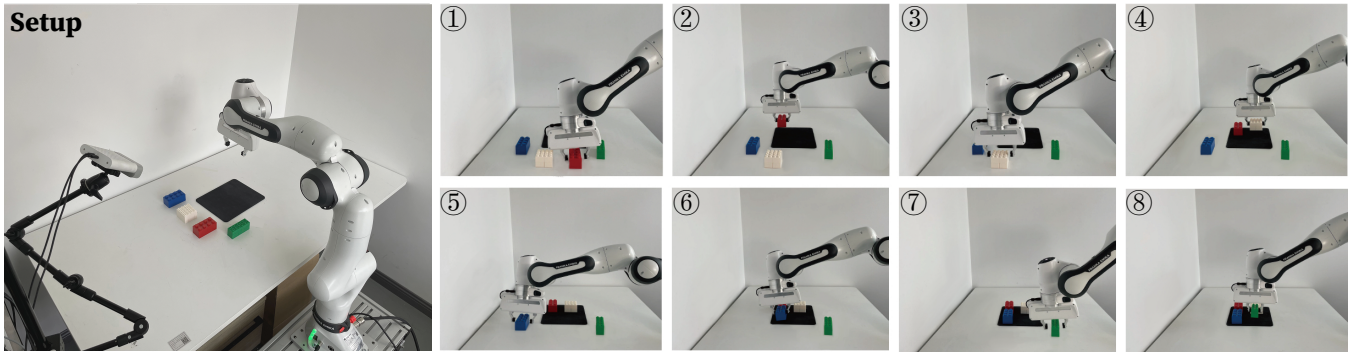

Fig. 6: **The real-world experiment on a physical robot.** The figure to the left shows the box-packing task setup. Actions ① to ⑧ are proposed by LLM^3 and successfully carried out by the physical manipulator.

REFERENCES

[1] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach," in *RSS*, 2016.

[2] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *RSS*, 2018.

[3] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, 2021.

[4] Z. Jiao, Z. Zhang, X. Jiang, D. Han, S.-C. Zhu, Y. Zhu, and H. Liu, "Consolidating kinematic models to promote coordinated mobile manipulations," in *IROS*, 2021.

[5] Z. Jiao, Z. Zhang, W. Wang, D. Han, S.-C. Zhu, Y. Zhu, and H. Liu, "Efficient task planning for mobile manipulation: a virtual kinematic chain perspective," in *IROS*, 2021.

[6] Z. Jiao, Y. Niu, Z. Zhang, S.-C. Zhu, Y. Zhu, and H. Liu, "Sequential manipulation planning on scene graph," in *IROS*, 2022.

[7] Y. Su, J. Li, Z. Jiao, M. Wang, C. Chu, H. Li, Y. Zhu, and H. Liu, "Sequential manipulation planning for over-actuated unmanned aerial manipulators," in *IROS*, 2023.

[8] 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 *ICRA*, 2016.

[9] 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 *IROS*, 2018.

[10] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *ICRA*, 2014.

[11] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[12] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning (ICML)*, 2022.

[13] 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," in *CoRL*, 2023.

[14] S. Mirchandani, F. Xia, P. Florence, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, A. Zeng, *et al.*, "Large language models as general pattern machines," in *CoRL*, 2023.

[15] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2020.

[16] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *International Journal of Robotics Research (IJRR)*, vol. 30, no. 6, 2011.

[17] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning.," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[18] Z. Yang, C. Garrett, T. Lozano-Perez, L. Kaelbling, and D. Fox, "Sequence-based plan feasibility prediction for efficient task and motion planning," in *RSS*, 2023.

[19] 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," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[20] H. Jiang, X. Yi, Z. Wei, S. Wang, and X. Xie, "Raising the bar: Investigating the values of large language models via generative evolving testing," *arXiv preprint arXiv:2406.14230*, 2024.

[21] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[22] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv:2305.16291*, 2023.

[23] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu, "Interpret: Interactive predicate learning from language feedback for generalizable task planning," in *RSS*, 2024.

[24] R. Gong, X. Gao, Q. Gao, S. Shakiah, G. Thattai, and G. S. Sukhatme, "Lemma: Learning language-conditioned multi-robot manipulation," *Robotics and Automation Letters (RA-L)*, 2023.

[25] R. Gong, Q. Huang, X. Ma, H. Vo, Z. Durante, Y. Noda, Z. Zheng, S.-C. Zhu, D. Terzopoulos, L. Fei-Fei, *et al.*, "Mindagent: Emergent gaming interaction," *arXiv preprint arXiv:2309.09971*, 2023.

[26] P. Ding, H. Zhao, Z. Wang, Z. Wei, S. Lyu, and D. Wang, "Quar-vla: Vision-language-action model for quadruped robots," *arXiv preprint arXiv:2312.14457*, 2023.

[27] W. Song, H. Zhao, P. Ding, C. Cui, S. Lyu, Y. Fan, and D. Wang, "Germ: A generalist robotic model with mixture-of-experts for quadruped robot," *arXiv preprint arXiv:2403.13358*, 2024.

[28] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[29] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *IROS*, 2023.

[30] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with llms as translators and checkers," *arXiv preprint arXiv:2306.06531*, 2023.

[31] J. Cui, T. Liu, N. Liu, Y. Yang, Y. Zhu, and S. Huang, "Anyskill: Learning open-vocabulary physical skill for interactive agents," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[32] P. Li, T. Liu, Y. Li, M. Han, H. Geng, S. Wang, Y. Zhu, S.-C. Zhu, and S. Huang, "Ag2manip: Learning novel manipulation skills with agent-agnostic visual and action representations," *arXiv preprint arXiv:2404.17521*, 2024.

[33] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum, "Predicate invention for bilevel planning," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[34] Y. Qian, P. Yu, Y. N. Wu, Y. Su, W. Wang, and L. Fan, "Learning concept-based causal transition and symbolic reasoning for visual planning," in *IROS*, 2024.

[35] X. Fang, C. R. Garrett, C. Eppner, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Dimsam: Diffusion models as samplers for task and motion planning under partial observability," *arXiv preprint arXiv:2306.13196*, 2023.

[36] 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 *RSS*, 2020.

[37] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[38] X. Gao, R. Gong, T. Shu, X. Xie, S. Wang, and S.-C. Zhu, "Vrkitchen: an interactive 3d virtual environment for task-oriented learning," *arXiv preprint arXiv:1903.05757*, 2019.

[39] X. Gao, R. Gong, Y. Zhao, S. Wang, T. Shu, and S.-C. Zhu, "Joint mind modeling for explanation generation in complex human-robot collaborative tasks," in *IEEE International Conference on Robot and Hhuman Interactive Communication (RO-MAN)*, 2020.

[40] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.