

Kinodynamic Task and Motion Planning using VLM-guided and Interleaved Sampling

Minseo Kwon and Young J. Kim

Abstract—Task and Motion Planning (TAMP) integrates high-level task planning with low-level motion feasibility, but existing methods are costly in long-horizon problems due to excessive motion sampling. While LLMs provide commonsense priors, they lack 3D spatial reasoning and cannot ensure geometric or dynamic feasibility. We propose a kinodynamic TAMP planner based on a *hybrid state tree* that uniformly represents symbolic and numeric states during planning, enabling task and motion decisions to be jointly decided. Kinodynamic constraints embedded in the TAMP problem are verified by an off-the-shelf motion planner and physics simulator, and a VLM guides exploring a TAMP solution and backtracks the search based on visual rendering of the states. Experiments on the simulated domains and in the real world show 32.14% \sim 1166.67% increased average success rates compared to traditional and LLM-based TAMP planners and reduced planning time on complex problems, with ablations further highlighting the benefits of VLM backtracking. More details are available at <https://graphics.ewha.ac.kr/kinodynamicTAMP/>.

I. INTRODUCTION

Robotic manipulation tasks, such as tabletop manipulations, require reasoning over both symbolic task decisions and continuous geometric feasibility. A robot must decide which action to perform—such as picking or placing—and which object to grasp, which constitutes a discrete search process. Simultaneously, it must determine grasp poses, feasible end-effector configurations, and collision-free motion trajectories governed by continuous constraints. This class of problems is studied under the framework of Task and Motion Planning (*i.e.*, TAMP), which combines high-level task planning with continuous action parameter binding and low-level motion planning [1].

When every high-level task plan has a feasible low-level motion solution, *i.e.*, the *downward refinement property*, task plans can be computed independently of geometric constraints and later refined with continuous action parameters. Unfortunately, this assumption rarely holds in practice, where geometric infeasibility can invalidate a task plan. Hence, most TAMP systems interleave discrete and continuous reasoning. Early approaches follow two main paradigms: *sequencing-first*, which generates symbolic task plans first and then attempts to satisfy continuous constraints, or *satisfaction-first*, which samples feasible motions first and integrates them into task planning [1]. The former suffers from repeated expensive constraint solving when samples are infeasible, while the latter wastes computation on many unused samples [2].

The authors are with the Department of Computer Science and Engineering at Ewha Womans University in South Korea {[minseo.kwon](mailto:minseo.kwon@ewha.ac.kr){kimy}@ewha.ac.kr.

With the emergence of large language models (LLMs), a new class of TAMP solvers has been proposed. These methods leverage the commonsense knowledge and high-level reasoning abilities of pretrained models to generate problem formulations for TAMP [3], plausible task sequences [4], and even continuous parameters, such as placement positions and motion failure reasoning [5]. However, due to the lack of their 3D spatial understanding, LLMs often struggle with precise motion reasoning and cannot reliably validate whether a plan is geometrically or physically feasible. Supplying them with high-dimensional numeric values like 6D poses or motion trajectories yields little benefit [4], [6]. Furthermore, existing TAMP approaches often ignore inertial or dynamic constraints in the planning environment. This motivates the need to incorporate kinodynamic constraints into our framework, ensuring that generated motions are executable in the real world.

To overcome these limitations, we propose a novel kinodynamic TAMP framework that interleaves a sequencing-first and a satisfaction-first approach, coupling task-level decisions and motion validations at every search-expansion step. This is abstracted through a *hybrid search tree* representation, where states and actions are grounded with sampled action parameters and immediately validated in physics simulation. An off-the-shelf motion planner and a physics simulator ensure kinodynamic feasibility, such as *e.g.*, collisions, kinematics, grasp stability, and object stability. The VLM further evaluates resulting world states to bias exploration toward promising branches and enable backtracking to recover from unexpandable states. We validate our approach on two simulated and one real-world manipulation domains with varying problem complexity. Our method outperforms a traditional domain-independent TAMP planner [7] and an LLM-based TAMP planner [5] in terms of success rates and achieves shorter planning times on complex problems, whereas the baselines mostly fail. We also perform an ablation study to show that VLM guidance significantly improves the quality of backtracking decisions.

Our contributions are summarized as follows:

- We introduce a *hybrid state tree* that unifies symbolic task decisions and instantiating continuous action, *i.e.*, a novel *interleaved* formulation of task and motion planning.
- To realize this abstraction, we combine a top- k symbolic planner for task diversity with a motion planner and a physics simulator, allowing us to check kinodynamic planning constraints.
- We demonstrate that VLMs can be effectively used not

only as a forward search heuristic but also for backtracking guidance, improving recovery from failures.

- We evaluate our framework on two simulated manipulation domains, achieving 32.14% \sim 105.56% increased average success rates in the Blocksworld domain and 280.00% \sim 1166.67% increased average success rates in the Kitchen domain compared to traditional TAMP and LLM-based planners, and provide ablation studies highlighting the impact of VLM guidance.
- We show that our TAMP planner works for a physical robot with kinodynamic constraints, close to the simulated results.

II. RELATED WORKS

A. Task and Motion Planning (TAMP)

Traditional TAMP approaches are commonly categorized as sequencing-first, satisfaction-first, or interleaved. Sequencing-first methods first compute a symbolic task plan and then attempt to satisfy continuous constraints. Repeatedly solving constraint satisfaction problems after motion failures can be costly: these methods may suffer from planning-time explosion for long horizon tasks [7], require workspace discretization that limits scalability [8], or rely on manually specified constraints that are difficult to generalize across different problems and domains [2]. Satisfaction-first methods [9], [10] generate samples before symbolic search, effective when sampling is cheap. That said, they suffer from combinatorial sample explosion, often producing many unnecessary samples.

Interleaved methods integrate sequencing and action parameter sampling within a unified search. For instance, sequencing-first planning can be combined with interleaved reasoning by pruning geometric samples that make the task plan infeasible [11]. However, this is limited to refining a single fixed plan skeleton and does not explore diverse task plans. More recent frameworks, such as [12], treat each plan skeleton as a top-level branch and employ MCTS for motion binding, but remain limited by random rollout policies that scale poorly in large domains. Our approach generates a discrete state graph to explore diverse task plans and expands the hybrid tree only with actions that satisfy all constraints, validated by a physics simulator for kinodynamic feasibility.

Learning-based approaches learn symbolic operators [13], or predicates [14] for TAMP. Recent works apply transformers [15] to predict the feasibility of plans or GNNs [16] to predict the relevance of sampled objects. However, data collection is costly, and generalization is limited, whereas we leverage pre-trained VLMs with commonsense knowledge.

B. LLMs/VLMs for Task and Motion Planning

LLMs and VLMs have recently been applied to robotic planning, leveraging commonsense reasoning to sequence high-level action primitives [17]–[19] or generate robot executable code that captures spatial reasoning and feedback loops [20]. They have also been combined with symbolic planning languages [21]–[24]. However, these methods are

restricted to task-level reasoning, without handling complex geometric or physical constraints.

LLMs and VLMs have been extended beyond simple task planning or skill sequencing to address TAMP problems. [25] employs LLMs to infer spatial relationships between objects, which are then passed to a TAMP planner. Other works generate problem formulations with LLMs/VLMs that are solved by external TAMP planners, with corrective re-prompting applied when failures occur [3], [26]–[28]. Similarly, [29] translates natural language constraints into Python APIs for an external TAMP planner, while [30] uses VLMs to infer open-world constraints—including both high-level action orders and continuous feasibility checks—through executable code integrated with an external planner. However, these approaches typically remain limited to coupling LLMs/VLMs with traditional TAMP planners.

[31] frames the search for valid LLM-generated Python programs and their continuous parameters as a constraint satisfaction problem. When violations occur, it enforces geometric and physical constraints through iterative re-prompting. LLMs can also directly produce task and motion plans: [4] generates batches of task plans refined with continuous parameters and expanded via MCTS, and [5] jointly generates symbolic sequences and continuous parameters, with re-prompting guided by motion planning feedback. However, unlike our approach, they rely solely on textual feedback for re-prompting and do not use visual cues to evaluate complex kinodynamic constraints during planning and replanning.

C. Kinodynamic Planning

Kinodynamic motion planning was first introduced by [32] to address both kinematic constraints, such as obstacle avoidance, and dynamic constraints, including limits on velocity, acceleration, force, and torque, to find time-optimal motion paths. [33] later proposed a randomized approach that extended RRTs to kinodynamic settings. Kinodynamic approaches have also been applied in TAMP. [34] integrated a sampling-based motion planner with a symbolic planner to jointly solve high-level task reasoning and low-level kinodynamic motion, thereby generating dynamically feasible trajectories. [35] employed differentiable physics engines to formulate planning as a gradient-based optimization problem, enabling reasoning over tool-use domains such as hitting and throwing. While classical kinodynamic planners focus on explicitly generating kinodynamic trajectories under robot dynamics, we instead focus on incorporating kinodynamic constraints and physical stability into the TAMP problem.

III. PROBLEM FORMULATION

We formulate our fully observable task and motion planning problem by extending the formulation in [36]:

$$P \equiv \langle \mathcal{H}, \mathcal{O}, \mathcal{A}, \mathcal{T}, h_0, \mathcal{G} \rangle. \quad (1)$$

Here, $\mathcal{H} = \mathcal{S} \times \mathcal{X}$ denotes the *hybrid state space*, and \mathcal{O} is a finite set of environment objects. \mathcal{S} is a finite set of symbolic states, where each symbolic state $s \in \mathcal{S}$

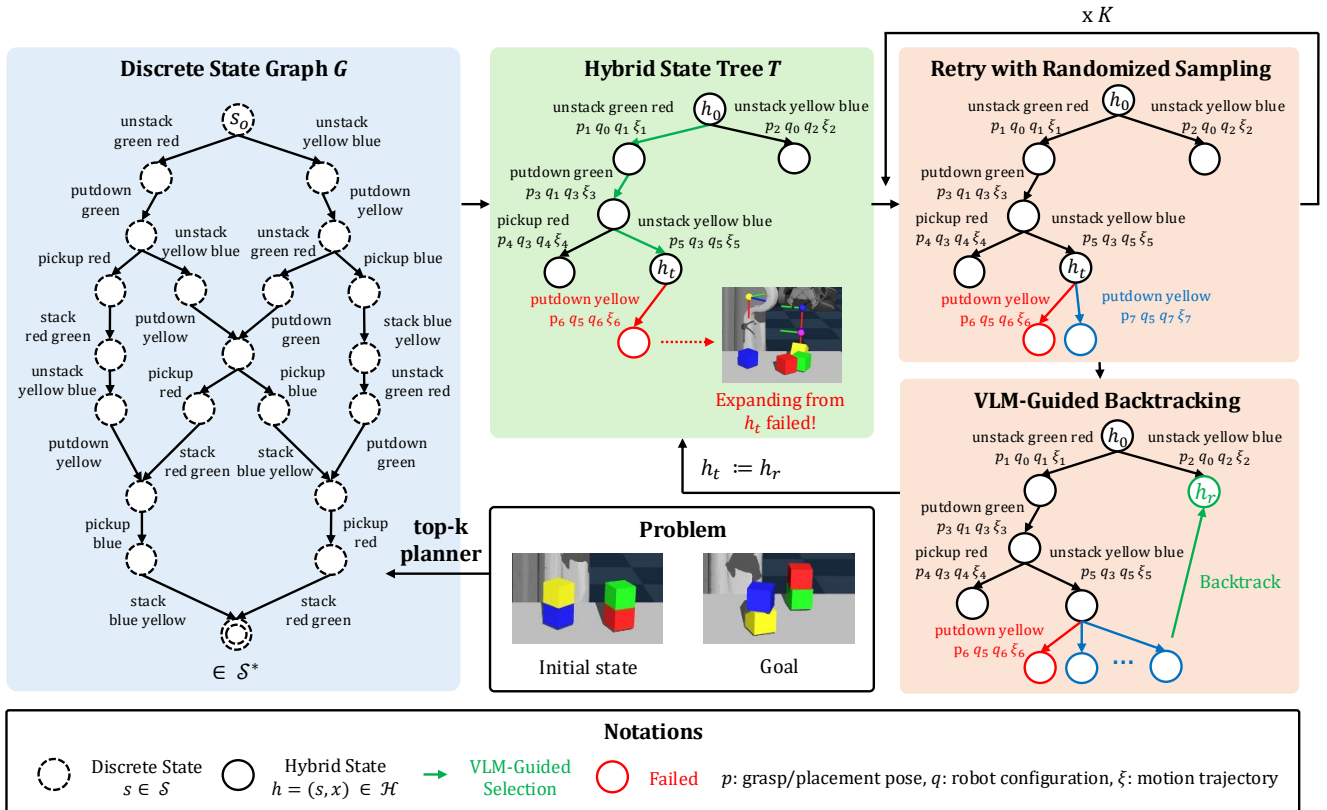


Fig. 1: An overview of our approach. Given a problem PDDL and a domain PDDL, a top- k symbolic planner generates a discrete state graph G . Guided by G , we expand a hybrid state tree T where each edge is expanded by motion planning and validated by physics simulation. If a node h_t fails to expand, we retry random sampling up to K times; if still unsuccessful, we prompt the VLM to predict a backtrack node h_r , from which expansion resumes. This process repeats until a goal is found or the timeout is reached.

consists of PDDL [37] predicates such as (on bacon stove). \mathcal{X} denotes the set of continuous world states (e.g., object poses, robot joint states, etc.). If $h = (s, x) \in \mathcal{H}$, then x must be *consistent* with s , meaning that the continuous state x satisfy the symbolic predicates in s . For instance, if $s = \{(\text{clear red}), (\text{on red green}), (\text{on-table green}), (\text{arm-empty})\}$, then in x , the red block must not have any object on top, its position must be located on top of the green block, the green block must be positioned on the table in a physically valid configuration, and no object should have contact with the arm gripper.

\mathcal{A} denotes the set of possible actions. An action $a(\delta, \kappa) \in \mathcal{A}$ is defined by an action name a (e.g., pickup, putdown in the PDDL domain), discrete parameters δ referring to objects and regions, and continuous parameters κ like goal pose, robot configurations and motion trajectory. $\mathcal{T} : \mathcal{H} \times \mathcal{A} \rightarrow \mathcal{H}$ is the state transition function, and $h_0 = (s_0, x_0) \in \mathcal{H}$ is the initial state. The set of goal states is defined as

$$\mathcal{G} = \{(s, x) \in \mathcal{H} \mid s \in S^*, x \in X(s)\}, \quad (2)$$

where $S^* \subset S$ is the set of symbolic goal states, and $X(s) \subset \mathcal{X}$ denotes the set of continuous states consistent with s that satisfy the symbolic goal conditions.

We assume the availability of samplers, i.e., motion planners, that propose candidate parameters κ , and a physics

simulator serving as the transition model \mathcal{T} . Our planning objective is to find a policy $\pi = \{a_1, \dots, a_n \mid \forall a_i \in \mathcal{A}\}$ for P in Eq. 1, such that successive transition from h_0 reaches a goal state $\exists h_n \in \mathcal{G}$ within a finite time of n .

IV. METHODS

An overview of our method is illustrated in Fig. 1.

A. Skeleton Space Generation

Sequencing-first methods compute a complete symbolic task plan as a skeleton, then attempt to refine it by solving a motion-level constraint satisfaction problem, and find a new skeleton if unsolvable. The chosen plan can be geometrically or physically infeasible—for example, an obstacle may block the path to the target object, or the target object may be in a cluttered environment, making it difficult to grasp. In order to recover from such dead ends, a TAMP planner must provide additional skeletons as alternatives. To address this, we adopt a top- k symbolic planner [38] to generate diverse symbolic task plans and organize them into a *discrete state graph* G (the blue block in Fig. 1). This graph provides a structured representation of the sampled skeleton space, enabling us to explore alternative task-level decisions without restarting the symbolic planner whenever motion refinement fails.

The top- k symbolic planner implements the K^* algorithm [39] on the Fast-Downward planner [40], a PDDL-based

symbolic planner. The objective of top- k symbolic planning is, given a domain PDDL and a problem PDDL, to compute $\Pi = \{\pi_1, \pi_2, \dots, \pi_k\}$, a set of k distinct plans of lowest cost that achieve the PDDL goal set S^* from the initial state s_0 .

Collectively, the set Π induces a directed graph $G = (V, E)$, which we call a *discrete state graph*:

- Each node $s \in V$ corresponds to a symbolic state represented as a set of PDDL predicates.
- Each edge $(s \xrightarrow{a} s') \in E$ corresponds to a grounded symbolic action a that transitions state s to state s' .

This discrete graph is not only a compact representation of the symbolic search space but also serves as a guide for hybrid state tree expansion. As explained in the next section, at each hybrid node, the outgoing edges of the corresponding discrete node define the set of admissible symbolic actions, which are then concretized with continuous parameters.

B. Hybrid State Tree Expansion

In the previous section, we constructed a discrete state graph G using top- k symbolic planning. This directed graph represents discrete states as nodes and discrete actions as edges, where a single state can have multiple parents. In contrast, two hybrid state nodes may share the same symbolic state but are extremely unlikely to have identical continuous states (e.g., object poses, robot configurations), since the continuous space is uncountable. Consequently, when planning in the hybrid state space, it is more natural to represent the search structure as a tree rather than a graph.

The root node of the hybrid state tree T , $h_0 = (s_0, x_0)$, is defined by the initial PDDL state s_0 specified in the problem PDDL and an initial continuous state x_0 . x_0 is instantiated by randomly sampling object states that satisfy the initial symbolic predicates s_0 . Let the current hybrid node be $h_t = (s_t, x_t)$. Expanding from this node proceeds in three stages (the green block in Fig. 1):

1) *Candidate Action Generation*: We first locate the corresponding discrete node s_t in the discrete state graph G and retrieve its outgoing edges, which represent an applicable set of symbolic actions $\{a_t^1(\delta^1), \dots, a_t^m(\delta^m)\}$ for s_t . These discrete actions are then refined into instantiated hybrid actions by sampling continuous parameters: $\{a_t^1(\delta^1, \kappa^1), \dots, a_t^m(\delta^m, \kappa^m) \mid \kappa^i = (p^i, q^i, q'^i, \xi^i)\}$ where $p^i \in \text{SE}(3)$ is an end-effector's pose in the workspace (e.g., grasp or placement pose) corresponding to the action a , $q^i \in \mathcal{C}$ is the current robot configuration in the configuration space \mathcal{C} and $q'^i \in \mathcal{C}$ is the robot configuration for p^i , and $\xi^i : [0, 1] \rightarrow \mathcal{C}$ is a motion trajectory from q^i to q'^i . The refinement process consists of:

- (a) **Goal sampling**: For pickup actions, a grasp pose is sampled based on the object's bounding box, with a randomly sampled roll angle applied to the end-effector frame. For putdown actions, a feasible placement position within the target region and the roll angle of the end-effector frame are sampled randomly. Then collision detection is handled by an analytic narrow-phase solver for box-box collisions after an AABB-based sweep-and-prune broad phase by [41].

(b) **Inverse kinematics (IK)**: For each sampled goal pose p^i , an IK solver computes a feasible robot configuration q^i .

(c) **Motion planning**: Given the goal configuration q'^i , a motion planner computes a collision-free trajectory from q^i to q'^i . We use the RRT-Connect planner [42].

If the IK solver fails to return joint positions yielding the desired end-effector pose, or if the motion planner fails to compute a collision-free trajectory, we regard this as a violation of *kinematic or collision constraints*. When such a violation occurs during the refinement of an action candidate $a_t^i(\delta^i, \kappa^i)$, the action is considered *infeasible*. If all candidate refinements fail, replanning is triggered.

2) *Candidate Action Simulation*: Each *feasible* candidate action $a_t^i(\delta^i, \kappa^i)$ is executed in the physics simulator, producing a successor hybrid state $h_{t+1}^i = \mathcal{T}(h_t, a_t^i(\delta^i, \kappa^i))$. During execution, the simulator records four exocentric views (front, top, left, right) of the scene for each resulting state. We employ [41] as the physics simulator, providing efficient physics simulation and photo-realistic rendering.

To verify inertial and contact dynamics effects from the robot motion, we check whether the resulting continuous state x_{t+1}^i obtained from the simulator is consistent with the symbolic state s_{t+1}^i (i.e., $x_{t+1}^i \in X(s_{t+1}^i)$). This ensures that the action outcome represents the intended symbolic effect without failures such as grasp slippage, collisions, or object collapse. For example, *grasp constraints* are violated if the object falls out of the gripper, while *release constraints* are violated if the object shows significant displacement after being released. If all candidates fail these checks, replanning is triggered.

3) *VLM-Guided Selection*: The set of valid successor states is $\{h_{t+1}^i \equiv (s_{t+1}^i, x_{t+1}^i) = \mathcal{T}(h_t, a_t^i(\delta^i, \kappa^i)), \forall i \mid a_t^i \text{ is feasible, } x_{t+1}^i \in X(s_{t+1}^i)\}$, where only actions that are both feasible and execution-consistent are included. This set is evaluated using a VLM, which leverages commonsense knowledge to guide the search. Specifically, we prompt the VLM with the simulator-rendered images of the current node h_t and candidate successor states, as well as the problem/domain descriptions. The VLM then selects the most promising successor state to continue the search toward the goal.

This hybrid expansion procedure ensures that symbolic decisions are immediately grounded in geometry and validated in simulation, while the VLM biases exploration toward states that are geometrically and kinodynamically consistent.

C. Replanning

We invoke a replanning mechanism when all action candidates are infeasible at h_t , or when every generated successor state is rejected due to kinodynamic constraint violations. In such cases, the planner attempts to recover through a two-stage strategy.

1) *Retry with Randomized Sampling*: First, we repeat the refinement process at h_t up to K times (the upper red block in Fig. 1). Both goal sampling and motion planning are randomized in candidate action generation, so repeated

trials increase the likelihood of success. Although a single invocation of RRT-Connect is not guaranteed to succeed, repeated iterations ensure probabilistic completeness. We fix $K = 5$ in our implementation.

2) *VLM-Guided Backtracking*: If all K retries fail, we invoke a VLM-guided backtracking strategy (the lower red block in Fig. 1). The VLM receives the simulator-rendered images of the current node h_t , the goal state, a JSON-encoded representation of the hybrid state tree expanded thus far, and a *constraint violation feedback* of previous expansion attempts. The JSON tree encodes each node with its name, discrete and continuous states, and each edge with information such as action labels, feasibility, and sampled parameters.

The constraint violation feedback is constructed from the outcomes of goal sampling, IK solver, motion planning, and candidate action simulation. Inspired by [5] and [31], we classify the feedback into four categories: (i) when no feasible IK solution exists, (ii) when the goal configuration collides with other objects, (iii) when the motion planner fails to produce a collision-free trajectory, and (iv) when a trajectory is found but grasp or release constraints are violated during execution. This structured feedback, together with the images of the current node h_t , helps the VLM identify the cause of failure using both textual feedback and visual cues. Based on this information, the VLM then selects an appropriate node to backtrack to within the JSON tree, say $h_r = (s_r, x_r)$. From h_r , step IV-B.1, step IV-B.2, and step IV-B.3 in Sec. IV-B are repeated until either a goal state $h_g \in \mathcal{G}$ is reached or the timeout is exceeded.

By combining randomized retries with VLM-guided backtracking, our framework efficiently recovers from unexpandable nodes and explores promising alternative states in the large hybrid search space. For example, suppose repeated attempts in the same region are likely to fail, such as in a cluttered workspace or blocked access. In that case, the VLM can backtrack to a node that enables clearing or repositioning other objects first.

V. EXPERIMENTS

A. Experimental Setup

All TAMP experiments were conducted on a workstation with a 13th Gen Intel Core i9-13900K CPU and an NVIDIA GeForce RTX 4090 GPU. We employed GPT-4o¹ from OpenAI as the VLM, with temperature fixed to 0.0. For generating symbolic plan skeletons, we used a top- k symbolic planner [38], with the number of plans fixed to $k = 30$. Genesis [41] was used as the physics simulator and photo-realistic renderer. A planning timeout of 600 seconds was imposed for all experiments, with planning proceeding until a goal was found or the timeout was reached. After the planning was terminated, the resulting simulator state was checked against the PDDL goal conditions; if satisfied, the trial was marked as a success; otherwise, a failure.

¹<https://openai.com>

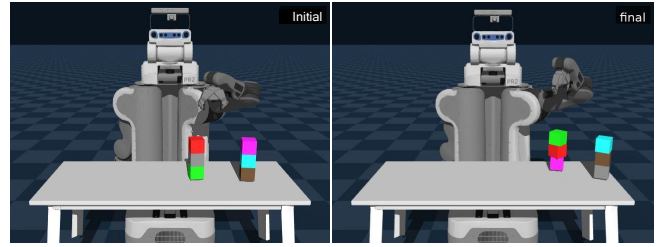


Fig. 2: Blocksworld domain setup with a PR2 robot. We only enable the left arm for this task. Initially, $n = 6$ blocks are randomly placed on a table (left), and the goal is to restack them in a new order (right).

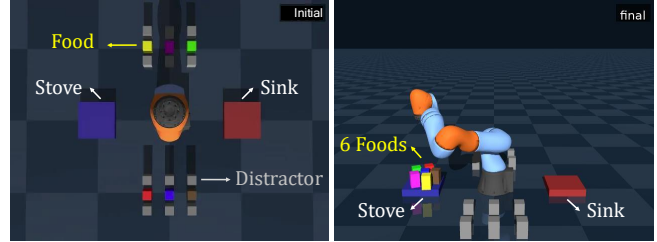


Fig. 3: Kitchen domain setup with a KUKA IIWA robot. Initially, six food objects are surrounded by 12 distractors (left), and the goal is to cook the food on the stove (right). The distractors are fixed objects.

We evaluated our approach in two simulated domains with varying numbers of objects n , which determine problem complexity:

- **Blocksworld**: This domain is based on the PDDL benchmark domain [43] from the International Planning Competitions (IPC). As in Fig. 2, n blocks are randomly divided into 2-3 stacks on the table, and the goal is to restack them in a new target order. Available actions include `pickup`, `putdown`, `stack`, and `unstack`.
- **Kitchen**: This domain is a modified version of the Kitchen domain used in PDDLStream [7]. As in Fig. 3, it contains 6 colored food objects (radish, egg, bacon, chicken, celery, apple), a sink (red cuboid), and a stove (blue cuboid). A food block becomes `Cooked` only after being cleaned on the sink and then cooked on the stove. The goal is to make a randomly chosen n food objects into `Cooked` state. To increase problem difficulty, we added gray distractor blocks around the initial positions of the food objects.

B. Performance Analysis

For both domains, we varied $3 \leq n \leq 6$. For each value of n , we randomly generated 10 problem instances and measured both success rate and planning time. Table. I shows the results for each domain. We quantitatively compared 4 approaches:

- 1) **Ours**: Our proposed method with VLM-guided backtracking, as introduced in Sec. IV-B.
- 2) **Ours w/o VLM Backtrack**: Our method without VLM guidance for backtracking. When a failure occurs, we return to a simple BFS-based strategy that

Domain		Blocksworld				Kitchen			
n		3	4	5	6	3	4	5	6
Ours	SR (%)	100	90	100	80	90	100	90	100
	Planning Time (s)	67.4221	98.1700	189.3949	267.3389	85.6074	93.5570	108.3262	120.2675
Ours w/o VLM Backtrack	SR (%)	100	80	70	50	90	80	80	100
	Planning Time (s)	55.4456	108.0900	170.0252	254.2184	80.4048	95.8650	110.1202	119.0159
PDDLStream	SR (%)	90	60	30	0	30	0	0	0
	Planning Time (s)	0.8247	14.0807	162.3950	Timeout	43.6913	Timeout	Timeout	Timeout
LLM ³	SR (%)	100	80	70	30	90	0	10	0
	Planning Time (s)	6.4879	74.0670	132.2868	270.5547	84.9920	Timeout	67.7273	Timeout

TABLE I: Average success rates (%) and planning times (s) of all baselines for $3 \leq n \leq 6$. Planning times are averaged over successful trials only. For cases with 0% success, the planning time is reported as "Timeout".

selects the first unvisited discrete node and resumes search from the corresponding hybrid node.

- 3) **PDDLStream** [7]: A domain-independent traditional TAMP baseline that integrates symbolic planning with black-box sampling functions (*e.g.*, grasp sampler, IK solver, RRT-Connect motion planner, etc.). We compare against its "adaptive" algorithm. Search-sample ratio is fixed to 1.
- 4) **LLM³** [5]: An LLM-based TAMP baseline where the LLM directly predicts symbolic action sequences and continuous parameters. We compare against the variant "LLM³ Backtrack", where the LLM selects a previous action for backtracking and replans from that point upon failure.

Comparisons: Our method outperforms baselines, achieving an average success rate of 92.5% in Blocksworld and 95% in Kitchen. In contrast, PDDLStream reports success rates of only 45% and 7.5% in Blocksworld and Kitchen, respectively. Moreover, its planning time grows exponentially as n increases, eventually leading to timeouts at $n = 6$ in Blocksworld, $n = 4, 5, 6$ in Kitchen for all ten problem instances. PDDLStream employs streams, which are black-box sampling functions that generate continuous parameters and certify constraints, such as collision-free motion. The outputs of streams, called "stream objects", are added to the symbolic state and reused when searching for new plans. As more stream objects accumulate, the planning state space grows significantly, slowing the process and often leading to inefficiency [2].

LLM³ achieved average success rates of 70% in Blocksworld and 25% in Kitchen, but suffered timeouts at $n = 4, 6$ in Kitchen for all problem instances and at $n = 5$ for 90% of problem instances. In Blocksworld, the success rate drops at $n = 6$, where the LLM increasingly predicts misordered stacking sequences, resulting in an average of 42 backtrack trials. In Kitchen, failures arise from collisions when placing food blocks on the stove, as LLMs struggle to predict collision-free poses and often continue producing infeasible continuous action parameters outputs even after replanning.

The comparison with the baselines clarifies the role of each component in our pipeline. The discrete state graph generated by the top- k planner explicitly bounds the symbolic search

space, preventing the planning-time explosion caused by uncontrolled search space growth observed in PDDLStream. Moreover, hybrid state tree expansion with motion planning and physics-based rollout grounds each action in geometric and dynamic feasibility, mitigating the geometric reasoning limitations of purely LLM-based TAMP planners.

Ablation Study of VLM-guided Backtracking: We conducted an ablation study to evaluate the effectiveness of VLM-guided backtracking. As shown in Table I, our method with VLM-guided backtracking achieved 23.33% and 8.57% increases in average success rates in Blocksworld and Kitchen, respectively, compared to the version without VLM guidance. This is because VLMs leverage common-sense knowledge and visual reasoning to identify the cause of failure in the current state and backtrack to a feasible node within one or two attempts. In contrast, without VLM-guided backtracking, the planner often repeatedly attempted backtracking until reaching a timeout. While incorporating VLM-guided backtracking slightly increased planning time due to additional VLM calls, the overall difference was marginal.

Large Task Space vs. Large Motion Space: Blocksworld exemplifies a *large task space*, as its complexity arises from the high branching factor induced by arbitrary block stacking configurations and discrete actions such as `stack` and `unstack` [16], [44]. By contrast, the Kitchen domain is characterized by a *large motion space*, where tight manipulation constraints make it challenging to grasp food objects in cluttered environments with distractors and to place multiple items on the stove or sink without collisions [12], [44].

Our ablation also confirms this distinction: VLM-guided backtracking yields a larger performance gain in Blocksworld than in Kitchen. In Blocksworld, where difficulty is primarily due to symbolic branching, VLM backtracking effectively explores alternative discrete states and mitigates task-level search complexity. In the Kitchen, however, the primary challenge lies in continuous motion feasibility under cluttered initial positions and constrained placements, so backtracking to a different discrete action has less impact. This demonstrates that VLM-guided backtracking is particularly effective in domains characterized by large task spaces, where symbolic branching dominates overall complexity.

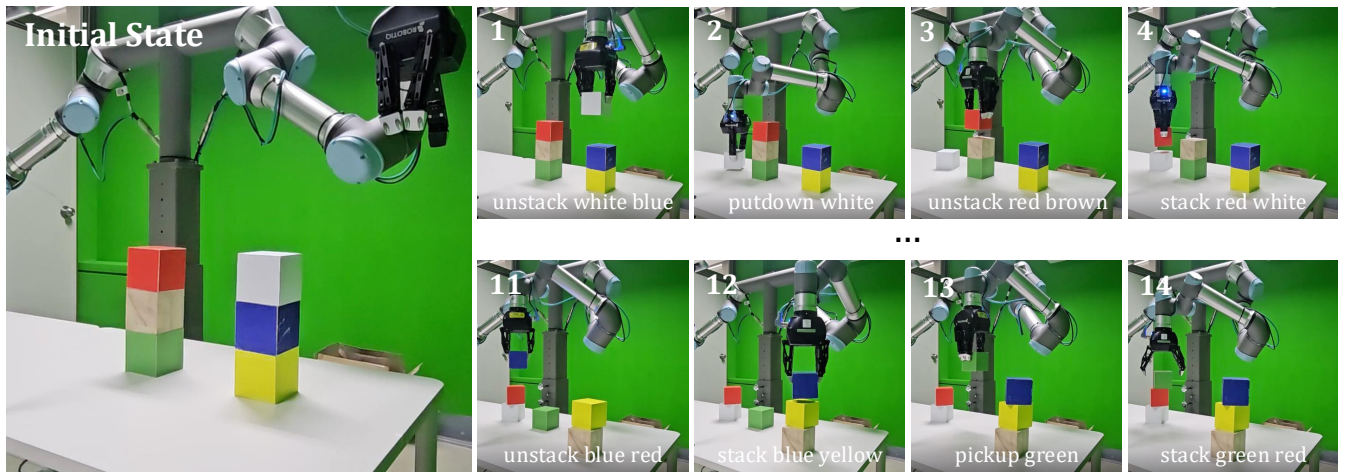


Fig. 4: Robotic demonstration of our TAMP planner in the Blocksworld domain ($n = 6$). The initial configuration consists of six blocks stacked on the table (leftmost image), and the goal is to rearrange them into the stacking sequence shown in 14. Only the first four and last four actions, $\pi = \{a_1, a_2, a_3, a_4, \dots, a_{11}, a_{12}, a_{13}, a_{14}\}$, are shown here.

C. Real-World Demonstration

We further validated our approach in Blocksworld through a real-world robotic demonstration. The setup consisted of dual UR5e manipulators equipped with Robotiq 3F grippers and an Intel RealSense D455 sensor. Given an RGB image of the tabletop scene captured by the sensor, an open-vocabulary object detection model [45] generated segmented masks for each object. These 2D masks were projected into 3D points using depth values, from which each object was localized in the simulator. Since the geometric and inertial properties of the objects were assumed to be known a priori, planning was performed as described in Sec. IV. The resulting plan, consisting of the arm trajectories and gripper open/close actions, was directly executed on the real robot, as shown in Fig. 4.

We conducted 5 trials for each $3 \leq n \leq 6$ in the Blocksworld domain. The planning times and success rates are similar to the simulation results presented in Sec. V. In particular,

- For $n = 3$ and $n = 4$, the success rates are 100%.
- For $n = 5$, two trials failed due to collisions between the gripper and the object, due to the inaccurate object localization caused by occlusion.
- For $n = 6$, the success rates are 80%, similar to the simulation.

These results demonstrate that our planning approach can generate feasible plans for long-horizon tasks applicable in real-world settings.

VI. CONCLUSION AND FUTURE WORK

This paper presents a novel kinodynamic TAMP planner that interleaves symbolic planning with kinodynamic feasibility checks using a physics simulator as the world transition model, while a VLM guides forward search and replanning.

Limitations: A major limitation of this work is that the computational overhead introduced by physics simulation

increases with plan length. Another limitation is that performance can be sensitive to the quality of the underlying samplers and variations in the VLM, such as different models, temperature settings, prompt design, and the number of input images. In addition, the assumptions of full observability and a black-box transition model remain restrictive.

Future Work: Future work will focus on extending our approach beyond pick-and-place-style manipulation to more diverse domains such as tool use, deformable objects, and contact-rich tasks. These domains require more expressive abstractions and validation strategies, and could be evaluated on more realistic benchmarks such as [46]. We can also reduce sensitivity to sampler quality by integrating learned sampling strategies to improve efficiency, as well as to relax assumptions such as full observability.

ACKNOWLEDGMENT

This work was supported in part by the ITRC/IITP Program (IITP-2026-RS-2020-II201460), and in part by the NRF (NRF-2022R1A2B5B03001385) in South Korea.

REFERENCES

- [1] 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*, vol. 4, no. 1, pp. 265–293, 2021.
- [2] B. Vu, T. Migimatsu, and J. Bohg, “Coast: Constraints and streams for task and motion planning,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 875–14 881.
- [3] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, “Autotamp: Autoregressive task and motion planning with llms as translators and checkers,” in *2024 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2024, pp. 6695–6702.
- [4] D. Lee, S. Joo, K. Lee, and B. Kim, “Prime the search: Using large language models for guiding geometric task and motion planning by warm-starting tree search,” *The International Journal of Robotics Research*, p. 02783649251347307, 2024.
- [5] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, “Llm³: Large language model-based task and motion planning with motion failure reasoning,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 086–12 092.

- [6] M. Sharma, "Exploring and improving the spatial reasoning abilities of large language models," *arXiv preprint arXiv:2312.01054*, 2023.
- [7] 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.
- [8] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.
- [9] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1796–1825, 2018.
- [10] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.
- [11] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson, "Efficiently combining task and motion planning using geometric constraints," *The International Journal of Robotics Research*, vol. 33, no. 14, pp. 1726–1747, 2014.
- [12] T. Ren, G. Chalvatzaki, and J. Peters, "Extended tree search for robot task and motion planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 048–12 055.
- [13] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, "Learning symbolic operators for task and motion planning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3182–3189.
- [14] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum, "Predicate invention for bilevel planning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 12 120–12 129.
- [15] Z. Yang, C. R. Garrett, T. Lozano-Pérez, L. Kaelbling, and D. Fox, "Sequence-based plan feasibility prediction for efficient task and motion planning," *arXiv preprint arXiv:2211.01576*, 2022.
- [16] M. Khodeir, B. Agro, and F. Shkurti, "Learning to search in task and motion planning with streams," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 1983–1990, 2023.
- [17] 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," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.
- [18] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.
- [19] 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.
- [20] 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," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.
- [21] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. Kaelbling, and M. Katz, "Generalized planning in pddl domains with pretrained large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, 2024, pp. 20 256–20 264.
- [22] T. Silver, V. Hariprasad, R. S. Shuttlesworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling, "Pddl planning with pretrained large language models," in *NeurIPS 2022 foundation models for decision making workshop*, 2022.
- [23] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.
- [24] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati, "Leveraging pre-trained large language models to construct and utilize world models for model-based task planning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 79 081–79 094, 2023.
- [25] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2086–2092.
- [26] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling, "Guiding long-horizon task and motion planning with vision language models," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 16 847–16 853.
- [27] J. Siburian, K. Shirai, C. C. Beltran-Hernandez, M. Hamaya, M. Görner, and A. Hashimoto, "Grounded vision-language interpreter for integrated task and motion planning," *arXiv preprint arXiv:2506.03270*, 2025.
- [28] M. Kwon and Y. J. Kim, "Neuro-symbolic task replanning using large language models," *The Journal of Korea Robotics Society*, vol. 20, no. 1, pp. 52–60, 2025.
- [29] W. Guo, Z. Kingston, and L. E. Kavraki, "Castl: Constraints as specifications through llm translation for long-horizon task and motion planning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 11 957–11 964.
- [30] N. Kumar, W. Shen, F. Ramos, D. Fox, T. Lozano-Pérez, L. P. Kaelbling, and C. R. Garrett, "Open-world task and motion planning via vision-language model inferred constraints," *arXiv preprint arXiv:2411.08253*, 2024.
- [31] A. Curtis, N. Kumar, J. Cao, T. Lozano-Pérez, and L. P. Kaelbling, "Trust the proc3s: Solving long-horizon robotics problems with llms and constraint satisfaction," *arXiv preprint arXiv:2406.05572*, 2024.
- [32] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.
- [33] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.
- [34] E. Plaku and G. D. Hager, "Sampling-based motion and symbolic action planning with geometric and differential constraints," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5002–5008.
- [35] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.
- [36] M. Kwon, Y. Kim, and Y. J. Kim, "Fast and accurate task planning using neuro-symbolic language models and multi-level goal decomposition," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 16 195–16 201.
- [37] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.
- [38] M. Katz, S. Sohrabi, O. Udrea, and D. Winterer, "A novel iterative approach to top-k planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, 2018, pp. 132–140.
- [39] H. Aljazzar and S. Leue, "K*: A heuristic search algorithm for finding the k shortest paths," *Artificial Intelligence*, vol. 175, no. 18, pp. 2129–2154, 2011.
- [40] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.
- [41] G. Authors, "Genesis: A generative and universal physics engine for robotics and beyond," December 2024. [Online]. Available: <https://github.com/Genesis-Embodied-AI/Genesis>
- [42] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.
- [43] J. Seipp, Á. Torralba, and J. Hoffmann, "PDDL generators," <https://doi.org/10.5281/zenodo.6382173>, 2022.
- [44] F. Lagriffoul, N. T. Dantam, C. Garrett, A. Akbari, S. Srivastava, and L. E. Kavraki, "Platform-independent benchmarks for task and motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3765–3772, 2018.
- [45] 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.
- [46] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44 776–44 791, 2023.