

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 COLLISION- AND REACHABILITY-AWARE MULTI-ROBOT CONTROL WITH GROUNDED LLM PLANNERS

Anonymous authors

Paper under double-blind review

ABSTRACT

Large language models (LLMs) have demonstrated strong performance in various robot control tasks. However, their deployment in real-world applications remains constrained. Even state-of-the-art LLMs, such as GPT-5, frequently produce invalid action plans that violate physical constraints, such as directing a robot to an unreachable location or causing collisions between robots. This issue primarily arises from a lack of awareness of these physical constraints during the reasoning process. To address this issue, we propose a novel framework that integrates reinforcement learning with verifiable rewards (RLVR) to incentivize knowledge of physical constraints into LLMs to induce constraints-aware reasoning during plan generation. In this approach, only valid action plans that successfully complete a control task receive positive rewards. We applied our method to two small-scale LLMs: a non-reasoning Qwen2.5-3B-Instruct and a reasoning Qwen3-4B. The experiment results demonstrate that constraint-aware small LLMs largely outperform large-scale models without constraint knowledge training, grounded on both the BoxNet task and a newly developed BoxNet3D environment built using MuJoCo, which involves LLM planning for up to 25 robots. This work highlights the effectiveness of grounding even small LLMs with physical constraints to enable scalable and efficient multi-robot control in complex, physically constrained environments. Our project website is at this link¹.

1 INTRODUCTION

Robotic control task requires controllers to find action plans given the robot’s physical constraints. Conventional methods often employ planning tools, such as PDDL (Fox & Long, 2003) and temporal logics (Emerson, 1990) to find optimal plans. However, they often demand expert knowledge to convert task constraints to formal language and struggle to scale efficiently in multi-robot systems due to increased search time (Chen et al., 2024a; 2025; Huang et al., 2022). Recent advances in Large Language Models (LLMs), which excel at complex reasoning tasks like math and coding (Luo et al.; DeepSeek-AI, 2025; Shao et al., 2024; Liu & Zhang, 2025), have inspired their application in robotic control. LLMs can interpret natural-language task instructions and generate valid action plans (Meng et al., 2025; Chen et al., 2024b; Chu et al., 2025); for instance, ChatGPT can effectively generate high-level commands such as “*Robot A, move the square object to panel 2*” (Chen et al., 2024b; Mandi et al., 2023). Paired with low-level execution functions that translate these commands into control signals for robots, they have proven successful in various multi-robot tasks (Chen et al., 2024b; Mandi et al., 2023; Sun et al., 2022).

However, these successes have mainly been in synthetic or constrained environments, where physical interactions are overly simplified. For example, most tasks in RocoBench have predefined all the possible valid robot interactions with the objects, largely restricting the action space for LLMs. This has led to significant issues in real-world scenarios, where LLM planners tend to violate many basic physical constraints. In particular, two important constraints are often overlooked. **Reachability constraint:** LLM would direct a robot arm to an unreachable position (Chen et al., 2024b; Zhang et al., 2025). **Collision constraint:** LLM would schedule robots to the same space, leading to collisions (Mandi et al., 2023; Jones et al., 2025).

¹<https://anonym-submission-user.github.io>

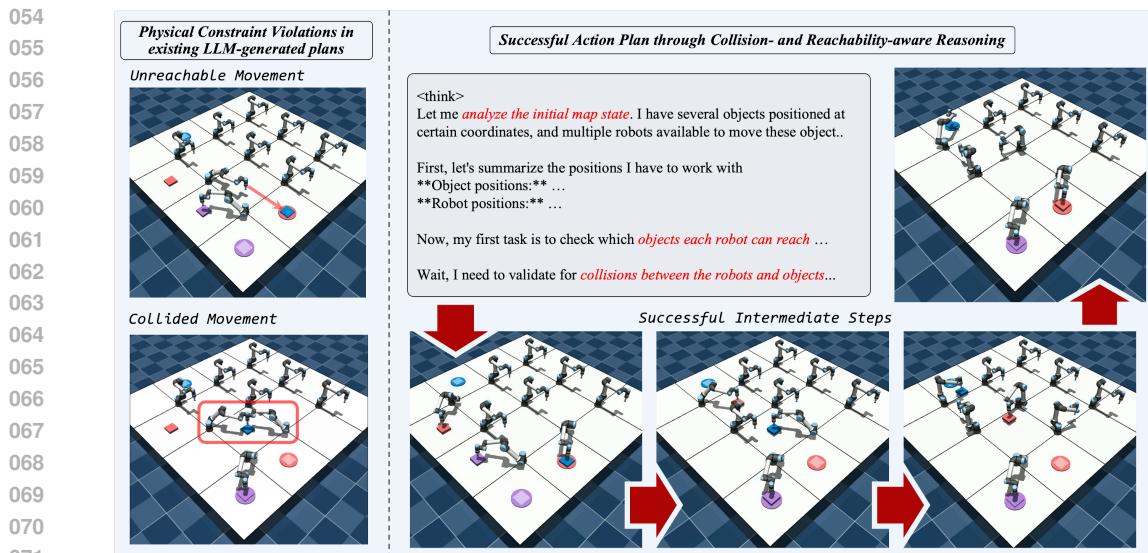


Figure 1: Illustration of LLM-based multi-robot control. (Left) Without grounding constraint knowledge, the LLM generates action plans that result in unreachable positions or collisions. (Right) Our planner generates valid movement actions through constraint-aware reasoning (highlighted in red) that successfully completes the BoxNet task after grounding robotic constraints knowledge.

As an example, Figure 1 (left) shows invalid actions generated by a SOTA reasoning LLM GPT-5, which easily violate these constraints, leading to significant safety and feasibility concerns.

These issues highlight the imperative to equip LLM planners with the ability to understand, analyze, and adhere to basic physical constraints. However, incorporating these constraints would require strong geometric reasoning and self-reflection capabilities, particularly when the number of robots is large, which may pose nontrivial challenges to LLMs. This raises a key research question: *Can LLMs, given their current reasoning capabilities, be trained to integrate physical constraints into the planning process? If so, to what extent can they succeed?*

To study these research questions, this paper presents a novel framework to incentivize these physical constraints into LLM planners, enabling them to reason about action validity during plan generation. Specifically, we leverage reinforcement learning with verifiable rewards (RLVR) that incorporates checks for reachability, kinematic feasibility, and collision avoidance. By using binary success-/failure signals derived from the robot control environment, we ensure that the LLM only receives rewards for generating physically valid plans. This fine-tuning process enables the LLM to reason about the validity during plan generation, leading to more reliable and collision-free action plans.

Our experiments on two LLMs, a non-reasoning Qwen2.5-3B-Instruct and a reasoning Qwen3-4B, have shown several encouraging findings. First, by incorporating the physical constraints into the reward, LLM planners can quickly acquire the ability to adhere to the physical constraints, thus drastically increasing the planning success rate, outperforming SOTA large-scale LLMs. For example, our best planner can achieve 0.87 and 0.53 pass@1 on two BoxNet-task multi-robot datasets that require controlling up to 25 robots, while the best baseline planner can only achieve 0.52 and 0.39 pass@1, respectively. Figure 1 (right) shows the thinking process and the generated plan by our fine-tuned LLM, which successfully solves the task without violating physical constraints. Second, our reasoning probing experiments have revealed that LLMs indeed learn to correctly identify whether the geometric constraints are satisfied or not. Finally, such capabilities acquired from RL can generalize to unseen environments with new environment size and geometric configurations, which further verifies that LLMs learn the generic reasoning skills rather than overfitting to specific geometric configurations.

In summary, the contributions of this work are as follows:

- We propose a novel framework that grounds LLMs with knowledge of action validity and collision constraints, ensuring LLM-planner-generated plans avoid unreachable positions, object collisions, or robot collisions.

- 108 • We introduce two new environments based on BoxNet task featuring tasks with up to
109 25 and 9 robots, respectively, which incorporate realistic physical constraints and serve as
110 testbeds for evaluating LLM-based multi-robot control.
- 111 • We implement our approach on two small-scale LLMs, demonstrating that even small mod-
112 els like Qwen2.5-3B-Instruct and Qwen3-4B, when grounded with physical constraints, can
113 outperform larger, state-of-the-art LLMs with no internal physical constraint knowledge in
114 complex multi-robot control tasks.

117 2 METHOD

119 2.1 OVERVIEW

121 In this section, we introduce our framework for grounding LLMs with reachability and collision
122 awareness. Denote $\mathcal{M}_{\theta_0}(\cdot)$ as the initial LLM for performing the robot control tasks, which is
123 capable of generating an action plan $\mathbf{s} \sim \mathcal{M}_{\theta_0}(\mathbf{q}; \mathcal{C})$ for solving the given control task described
124 by \mathbf{q} under a set of physical constraints \mathcal{C} , such as the reachability of a robot arm and collision
125 avoidance. Our goal is to fine-tune the LLM such that the generated solution \mathbf{s} successfully moves
126 objects to their target positions while not violating the constraints \mathcal{C} . In the following, we first
127 introduce the RLVR framework for grounding physical constraints in Section 2.2, then introduce our
128 initial LLM policy warmup strategy in Section 2.3, and two different planner modes we consider in
129 Section 2.4.

130 2.2 GROUNDING LLM WITH PHYSICAL CONSTRAINTS THROUGH RLVR

132 We adopt a similar RL framework to the DeepSeek-R1 LLM (DeepSeek-AI, 2025; Ren et al., 2025),
133 which employs the group relative policy optimization (GRPO) algorithm (Shao et al., 2024). Specif-
134 ically, at each training step $i, i \geq 1$, we sample a group of different plans and its corresponding
135 reasoning $\{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_G\}$ from the old LLM policy $\mathcal{M}_{\theta_{i-1}}$ for each query robot-control task \mathbf{q} ,
136 where G is the group size. Each plan \mathbf{s}_j is simulated in a manually implemented environment with
137 physical constraints. The corresponding reward function $r(\cdot)$ later estimates whether it successfully
138 completes the given task with the simulation environment feedback. Then the LLM is optimized by
139 maximizing the following objective (Shao et al., 2024).

$$\begin{aligned}
\mathcal{J}_{GRPO}(\mathcal{M}_{\theta_i}) &= \mathbb{E} \left[\left(\mathbf{q} \sim \mathcal{D}, \{\mathbf{s}_j\}_{j=1}^G \sim \mathcal{M}_{\theta_{i-1}}(\mathbf{O}|\mathbf{q}; \mathcal{C}) \right) \right] \\
&= \frac{1}{G} \sum_{j=1}^G \left(\min \left(\frac{\mathcal{M}_{\theta}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})}{\mathcal{M}_{\theta_{i-1}}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})} A_j, \text{clip} \left(\frac{\mathcal{M}_{\theta}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})}{\mathcal{M}_{\theta_{i-1}}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})}, 1 - \epsilon, 1 + \epsilon \right) A_j \right) \right. \\
&\quad \left. - \beta \mathbb{D}_{KL}(\mathcal{M}_{\theta_i} \parallel \mathcal{M}_{\theta_0}) \right),
\end{aligned}$$

147 where \mathcal{D} denotes the training data and A_j represents the advantage, computed as the reward of each
148 plan subtracted by the average reward within the group. Detailed definitions are in Appendix D.

149 Our reward function, denoted as $r(\cdot)$ largely follows the design in DeepSeek-R1 (DeepSeek-AI,
150 2025), with an additional plan efficiency term. Specifically,

$$r(\mathbf{s}; \mathbf{q}, \mathbf{s}^*, \mathcal{C}) = r_{\text{format}}(\mathbf{s}) + r_{\text{execute}}(\mathbf{s}; \mathcal{C}) - r_{\text{efficiency}}(\mathbf{s}; \mathbf{s}^*),$$

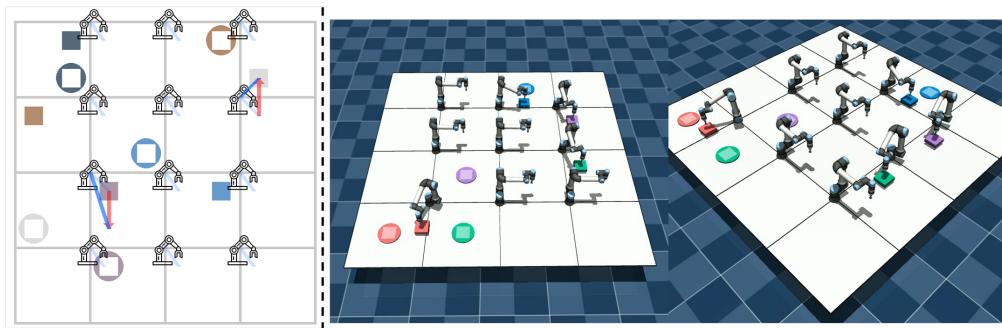
154 $r_{\text{format}}(\mathbf{s}) = 0.1$ if the generated solution adheres to the required thinking-then-response format and
155 0 otherwise. $r_{\text{execute}}(\mathbf{s}; \mathcal{C}) = 1$ if the simulator verifies that the plan ① accomplishes the task AND
156 ② no physical constraints are violated, and 0 otherwise. Incorporating physical constraint checking
157 in r_{execute} is the key mechanism to improve constraint awareness of the LLM planner. Finally,

$$r_{\text{efficiency}}(\mathbf{s}; \mathbf{s}^*) = \max(0, 0.1 \times (\text{len}(\mathbf{s}) - \text{len}(\mathbf{s}^*)))$$

160 penalizes the excessive length compared to a golden plan, \mathbf{s}^* , which is obtained by an A* search
161 algorithm (see Appendix D for search details). Additionally, we implement a minimum cap of
162 $2 \times r_{\text{format}}(\mathbf{s})$ when \mathbf{s} is a valid plan that successfully solves the task to ensure that correct plans
163 always receive a higher reward than incorrect ones.

162 Table 1: Example synthesized reasoning trace for FULLPLAN planner and REPLAN planner. We
 163 highlight three reasoning behaviors: 1. environment analysis in **blue**, 2. validity verification in
 164 **green**, and 3. efficiency consideration in **orange**. The environment feedback is marked in gray.
 165

165 FULLPLAN Planner Reasoning:	166 REPLAN Planner Reasoning:
166 Object positions: 167 Object 0: [0.75, 1.75] ...	167 <observation> 168 Object 0: [0.75, 2.75] ...
168 <think>Okay, let me analyze the given environment before coming up with a movement plan ...	169 </observation>
169 ### Evaluating Reachability: 170 - ***Robot 0*** cannot reach either object ...	170 <think>Alright, let's break down the situation ...
171 ### Collision Checks: 172 - No objects are placed at the same position... 173 Let me finalize this, since there are no collisions, 174 this plan efficiently moves all objects ...	171 <observation> 172 Object 1: [2.25, 2.75] ... 173 </observation> 174 <think>Alright, let's analyze this ...



175
 176 Figure 2: (Left) An example BoxNet2D environment. The blue lines mark the robot arm, and
 177 the red lines mark the movement trajectory. (Right) An example BoxNet3D environment. Both
 178 environments require robots to collaborate to move boxes to the circle with the corresponding color.
 179
 180

181 2.3 INITIAL SUPERVISED FINE-TUNING (SFT) WARMUP

182 Prior works have shown that LLMs' initial performance on a reasoning task is crucial to RLVR
 183 training (Luo et al.; Shao et al., 2024; Liu & Zhang, 2025). Since off-the-shelf LLMs often struggle
 184 with robotic control tasks, we introduce an SFT warmup to equip them with basic robot control
 185 knowledge before RL training.

186 The SFT data needs to contain two components: ① a correct plan to solve a given task, and ② a
 187 reasoning chain that reflects a multi-step decision-making process leading to the correct plan. To
 188 synthesize such data, for each task, we first use the A* search algorithm to search for the optimal
 189 plan. Then, we pass the plan to an LLM, which is prompted to generate a reasoning process for the
 190 plan, consisting of the following three patterns:

- 201 • *Analysis of the given environment*, where the LLM assesses the current positions of robot arms
 202 and objects, e.g., *let me analyze the current situation*;
- 203 • *Validity verification*, where the LLM reasons about an arm's reachable area based on its base po-
 204 sition and potential collision between different arms, e.g., *If Robot 0 moves, ..., it will collide*;
- 205 • *Efficiency considerations*, where the LLM evaluates whether multiple movements can be par-
 206 alelized to improve the plan efficiency.

207 Table 1 shows the example reasoning chain synthesized by GPT4o-mini, where the three patterns
 208 are rendered in different colors. Appendix F shows the full prompt for our reasoning synthesis.
 209

210 2.4 TWO PLANNERS: FULLPLAN PLANNER AND REPLAN PLANNER

211 We consider two different LLM-based planners in this work. The first planner, referred to as
 212 FULLPLAN, involves the LLM directly generating the entire plan that may take multiple execu-
 213 tion steps for solving a task based on the initial positions of all objects and robot poses in the
 214 environment. The second planner, denoted as REPLAN, generates one step at a time, observing the
 215 updated object positions from the environment (appended to its context) before generating the next

216 step. This allows the planner to evolve dynamically as the environment changes through multiple
 217 steps. Table 1 provides the example planning processes for two different LLM-based planners for
 218 the same initial environment. We highlight that the REPLAN planner observes multiple intermediate
 219 observations of the environment, while the FULLPLAN planner only sees the initial environment.
 220

222 3 BOXNET-BASED MULTI-ROBOT ENVIRONMENTS

224 In this work, we primarily experiment with BoxNet task (Chen et al., 2024b), where multiple
 225 robots collaborate to move objects across different cells to targeted locations in a fixed grid map.
 226 This section details two environments we developed, a modified BoxNet2D environment and the
 227 newly developed BoxNet3D environment, both equipped with realistic physical constraint checks.
 228

229 **BoxNet2D.** Figure 2 (left) shows a BoxNet2D environment. In this environment, robot arms
 230 are placed at a corner of a grid environment. Each arm can reach its neighboring grids for picking
 231 and placing objects. Unlike the previous BoxNet environment that predefined all valid robot arm
 232 actions, we allow LLMs to generate spatial coordinates directly, significantly expanding the action
 233 space. For example, the action “*Robot0 Move (1.25, 1.25) → (1.75, 1.75), False*” moves *Robot0*’s
 234 arm to (1.75, 1.75) without picking up an object. In contrast, “*Robot1 Move (2.25, 1.75) → (1.25,*
 235 *1.25), True*” indicates *Robot1* picking up the object at (1.75, 1.25) and moving it to (2.25, 1.75). **For**
 236 **BoxNet2D environment, each arm moves along a straight path from the start point to target point.**

237 We pre-define four points within each grid, *e.g.*, (0.25, 0.25), (0.25, 0.75), (0.75, 0.25), (0.75, 0.75),
 238 for object placement and robot arm moving, and later we will show that the fine-tuned LLM can gen-
 239 eralize to other points in experiments. Three physical constraints are incorporated: **1** *reachability*
 240 *verification*, which checks whether the target position of a robot is unreachable. **2** *robot collision*
 241 *detection*, which checks whether the movement trajectory of different arms intersects with each
 242 other, or one robot’s movement trajectory intersects with another robot arm, leading to a collision.
 243 **3** *object collision detection*, which checks whether two objects are placed at the same spatial co-
 244 ordinates during the plan execution. **To implement these constraints, we manually implement these**
 245 **constraint detection functions and raises error when any constraint is not satisfied.** For example,
 246 **reachability verification checks whether an arm’s position has larger offset to its base, *i.e.*, $\Delta x > 1$**
 247 **or Δy .** The robot collision constraint detection whether two arm movement paths has intersecting
 248 point. Example invalid actions of BoxNet2D are provided in Appendix E.

249 **BoxNet3D.** Figure 2 (right) shows a BoxNet3D environment. In this environment, we employ
 250 the UR5e robot arm as the basic robot arm². Similar to the 2D environment, the goal is to move
 251 colored boxes into corresponding circles of the same color with the fewest actions. **Each robot arm’s**
 252 **base is fixed at grid joint and moves its arm around to reach adjacent grid center position for object**
 253 **pickup and placement.** We employ an RRT planner implemented by RoCoBench (Mandi et al.,
 254 2023) for low-level control signal generation, *i.e.*, the robot joint configuration trajectory during
 255 each step of movement, given LLM planner-generated coordinates for arm position movement³. We
 256 employ Mujoco as the engine (Todorov et al., 2012) to power arm reachability check and collision
 257 detection, where unreachable position result in failed robot joint calculation in the low-level RRT
 258 planner, and collision result in geometry contact that can be captured by Mujoco engine. Example
 259 invalid actions of BoxNet3D are provided in Appendix E.

261 4 EXPERIMENT

263 In this section, we conduct empirical experiments on the two BoxNet-based environments to assess
 264 the effectiveness of our method. We first present the experiment setup in Section 4.1 and then the
 265 experiment results in Section 4.2, followed by additional ablation studies in Section 4.3.
 266

268 ²<https://www.universal-robots.com/products/ur5e/>

269 ³The RRT planner is adapted from the implementation in RoCoBench official code base (<https://github.com/MandiZhao/robot-collab/blob/main/rocobench/rrt.py>)

Table 2: **Performance of different LLM planners on BoxNet2D and BoxNet3D.** For each task, we report *Success*, ratio of pass@1 accuracy over four trials, *StepDiff.*, the difference in number of steps between model-generated plans and A* searched plans for successful executions, and *Para.*, the maximum number of robots operatin in parallel. For each model, the performance for FULLPLAN planner and REPLAN planner side-by-side (FULLPLAN / REPLAN).

Model	BoxNet2D			BoxNet3D		
	Success \uparrow	StepDiff. \downarrow	Para. \uparrow	Success \uparrow	StepDiff. \downarrow	Para. \uparrow
Search Algorithm						
A*	1	0	2.24	1	0	2.14
LLMs without constraint knowledge grounding						
GPT-4omini	0.06 / 0.05	2.35 / 0.14	1.17 / 1.15	0.07 / 0.06	0.79 / 0.45	1.03 / 1.08
GPT-4o	0.12 / 0.11	2.14 / 0.13	1.15 / 1.22	0.10 / 0.12	0.23 / 0.68	1.35 / 1.11
GPT-o4mini	0.37 / 0.35	0.24 / -0.31	1.58 / 1.87	0.11 / 0.33	0.14 / 1.21	1.45 / 1.53
GPT-5-mini	0.43 / 0.31	1.13 / 0.44	1.44 / 1.75	0.23 / 0.30	0.25 / 0.44	1.24 / 1.33
GPT-5	0.52 / 0.44	0.78 / 0.13	1.52 / 1.61	0.31 / 0.39	0.29 / 0.37	1.34 / 1.21
Qwen2.5-3B-Inst	0.0 / 0.0	— / —	— / —	0.08 / 0.0	0.20 / —	1.40 / —
Qwen2.5-7B-Inst	0.02 / 0.02	1.45 / 0.31	1.20 / 1.23	0.05 / 0.08	0.41 / 0.35	1.13 / 1.07
QwQ-32B	0.04 / 0.07	0.35 / 0.17	1.12 / 1.21	0.07 / 0.15	0.24 / -0.09	1.08 / 1.31
Qwen3-4B	0.14 / 0.13	0.23 / 0.14	1.29 / 1.29	0.15 / 0.11	0.07 / 0.31	1.17 / 1.14
Qwen3-8B	0.18 / 0.15	-0.23 / -0.34	1.24 / 1.31	0.17 / 0.13	-0.02 / 0.09	1.22 / 1.21
Qwen3-14B	0.19 / 0.21	-0.31 / -0.24	1.34 / 1.41	0.10 / 0.14	0.17 / 1.37	1.34 / 1.35
Qwen3-32B	0.11 / 0.14	0.17 / -0.03	1.24 / 1.12	0.14 / 0.17	0.09 / 0.04	1.27 / 1.09
LLMs with grounded constraint knowledge						
Qwen2.5-3B-SFT	0.34 / 0.30	0.11 / -0.04	1.51 / 1.39	0.27 / 0.39	-0.07 / -0.05	1.27 / 1.39
Qwen2.5-3B-RL	0.58 / 0.68	-0.65 / 0.23	1.53 / 1.50	0.42 / 0.48	-0.15 / -0.14	1.33 / 1.49
Qwen3-4B-SFT	0.45 / 0.31	-0.12 / -0.15	1.92 / 1.35	0.37 / 0.43	0.09 / -0.11	1.32 / 1.48
Qwen3-4B-RL	0.87 / 0.75	-0.84 / -0.64	1.73 / 1.64	0.45 / 0.53	-0.25 / -0.29	1.39 / 1.56

4.1 EXPERIMENT SETUP

Dataset generation. We create environments with various map sizes and object initial and target positions for both BoxNet2D and BoxNet3D. Specifically, for BoxNet2D, we use map sizes ranging from 2×2 to 6×6 and 1 to 5 objects, resulting in 55,000 training and 250 testing environments. For BoxNet3D, we use map sizes from 2×2 to 4×4 with 1 to 3 objects, yielding 1,800 training and 160 testing environments. The object position is randomly sampled from the pre-defined points, while the robot arms are evenly placed at the grid joints to ensure that all grids in the map can be reached. With the robot placement strategy introduced earlier, the maximum number of robots reaches up to 25 and 9 for BoxNet2D and BoxNet3D, respectively. For each randomly sampled environment, the manually implemented A* search algorithm verifies that a valid solution action plan exists. While our focus is on solvable environments, we include a discussion of unsolvable cases in Appendix C.4. Detailed dataset statistics are summarized in Appendix D.3.

Evaluation metric. We evaluate LLM-based planners mainly from two perspectives: ① *Success*, the proportion of generated plans that solve given robotic tasks, measured by *pass@1* over four trials per environment; and ② *StepDiff.*, the difference in number of steps between successful plans and the best plan among A* solutions. We also report *Para.*, the maximum number of robots operating in parallel in any intermediate step of a successful plan.

Baseline LLMs. We mainly compare with off-the-shelf LLMs via direct prompting on robot control task. To ensure comprehensive coverage of existing LLMs, our evaluation includes both reasoning and non-reasoning models, closed-source and open-source ones across different parameter scales. Specifically, we consider closed-source LLMs, GPT-4o, GPT-4o-mini, GPT-o4-mini, GPT-5-mini and GPT-5. On the open-source side, we include Qwen-2.5 and Qwen3 series, with parameter sizes ranging from 3B to 32B. We also compare with LLM-based symbolic translation approach such as AutoTAMP (Chen et al., 2024a) in Appendix C.1.

324 **Training details.** We use two base LLMs: a non-reasoning LLM Qwen-2.5-3B-Instruct and a
 325 reasoning LLM Qwen3-4B. For SFT warm-up, we use a learning rate of $1e - 5$ for Qwen-2.5-
 326 3B-Instruct and $3e - 5$ for Qwen3-4B with the AdamW optimizer (Loshchilov & Hutter, 2017).
 327 Training runs for 10 epochs on FULLPLAN and 5 epochs on REPLAN. RLVR training uses a fixed
 328 $1e - 6$ learning rate for 200 steps with the GRPO algorithm (Guo et al., 2025). Batch sizes are
 329 256 (group size 8) for BoxNet2D and 64 for BoxNet3D. Following prior work (Luo et al.; Liu &
 330 Zhang, 2025), we set $\beta = 0$ in the GRPO loss. We use the VeRL framework (Sheng et al., 2024), and
 331 run all experiments on 2×8 NVIDIA H100 GPUs. Detailed computation cost is in Appendix D.2.
 332
 333

334 4.2 EXPERIMENTAL RESULTS

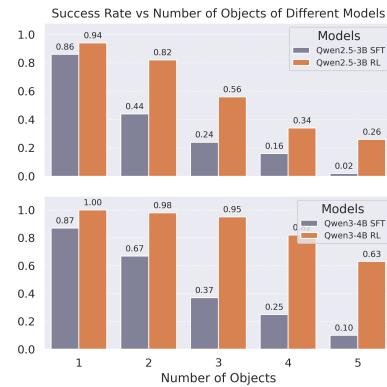
335 **Grounding empowers small-scale LLMs to outperform larger ones.** We first evaluate
 336 the grounded LLM planner performance in Table 2 and a detailed error type breakdown
 337 is in Appendix C.3. The LLMs with physical constraints knowledge grounded through
 338 SFT warmup and further RL training are denoted by the suffix **-SFT** and **-RL**, respectively.
 339 We highlight three observations: First, grounding constraint knowledge significantly boosts planning performance,
 340 enabling 3B and 4B LLMs to achieve higher success against larger ones. For example, Qwen3-4B-RL
 341 FULLPLAN planner achieves 0.87 success rate, 0.5 higher than the best baseline GPT-o4mini.
 342 Second, grounded LLM planners produce more efficient plans than the A*
 343 search algorithm on solved tasks. For example, Qwen3-4B-RL has 0.84 fewer steps than the ground-truth plan
 344 from our A* implementation, showing a strong reasoning ability and also echoes the findings in prior works
 345 that compare LLM planners with A* on Sudoku (Lehnert et al., 2024; Su et al., 2024). Third, planner performance
 346 on BoxNet3D is generally worse than on BoxNet2D. This suggests that, although we applied multiple feasibility
 347 checks in BoxNet2D, some physical constraints remain missing. The BoxNet3D environment uses a more
 348 advanced simulation engine and thus exposes more limitations. These results underscore the importance of developing realistic robotic environments that
 349 capture real-world complexity for future LLM-based robotic control research.
 350
 351

352 **Figure 3** visualizes performance against the number of boxes for BoxNet2D. We highlight that
 353 RL-trained planners better preserve performance when task complexity increases. For example, the
 354 performance gap between Qwen3-4B-SFT and Qwen3-4B-RL grows from 0.13 to 0.53 when the
 355 number of boxes increases from 1 to 5, highlighting better scalability of RL planners.
 356
 357

358 **RL planners generalize better to**
 359 **unseen environments.** To measure
 360 how the planners’ reasoning ability
 361 generalizes, we evaluate the
 362 planners’ performance in two unseen
 363 variants of BoxNet2D test data: ①
 364 Random robot layout, denoted as
 365 RANDROB, where the robot positions
 366 are randomly assigned on the grid
 367 joints in maps ranging from 2×2 to
 368 5×5 . We ensure that all testing data
 369 are solvable, which means every box
 370 can reach its target position via robot movement. ② Unseen coordinates, i.e., NEWCOORD, where
 371 the initial and target position coordinates of all objects in BoxNet2D test set are perturbed by a
 372 random offset $(\Delta x, \Delta y) \sim \mathcal{U}([-0.2, 0.2]^2)$. ③ Unseen map size, i.e., NEWMAP, where the testing
 373 grid size is extended to two unseen sizes 10×5 and 7×7 . Example data are shown in Appendix E.
 374

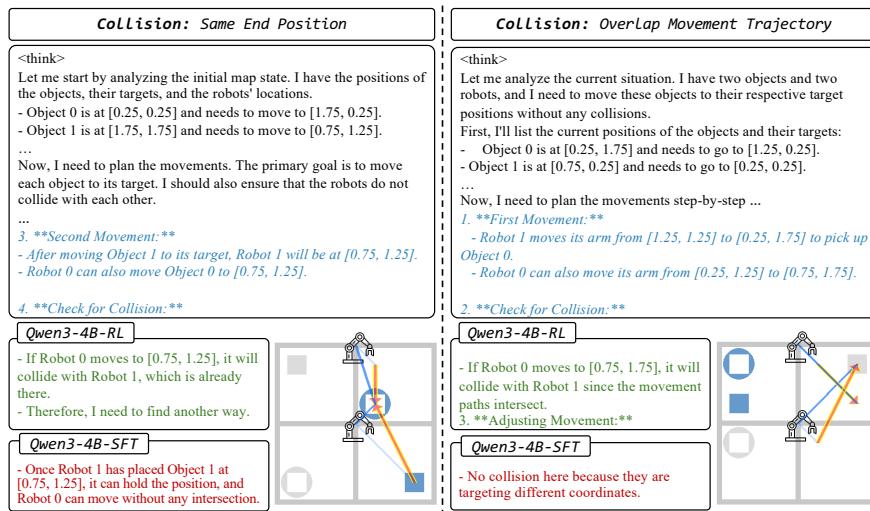
375 Table 3: Planning performance generalization on unseen
 376 BoxNet2D environments.

Model	RANDROB		NEWCOORD		UNSEENMAP	
	Success \uparrow	StepDiff. \downarrow	Success \uparrow	StepDiff. \downarrow	Success \uparrow	StepDiff. \downarrow
FULLPLAN Planner						
Qwen2.5-3B-SFT	0.39	0.12	0.32	0.21	0.25	0.17
Qwen2.5-3B-RL	0.58	-0.04	0.55	-0.32	0.35	0.05
Qwen3-4B-SFT	0.48	0.03	0.43	-0.03	0.30	0.11
Qwen3-4B-RL	0.79	-0.40	0.87	-0.39	0.40	-0.24
REPLAN Planner						
Qwen2.5-3B-SFT	0.39	0.23	0.33	0.09	0.18	0.22
Qwen2.5-3B-RL	0.71	1.24	0.68	1.04	0.39	0.14
Qwen3-4B-SFT	0.41	-0.03	0.37	0.15	0.32	-0.05
Qwen3-4B-RL	0.75	-0.15	0.69	0.09	0.42	-0.11



377 Figure 3: Success rate against number of boxes (up) and against in the
 378 BoxNet2D environment.

378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394



395 Figure 4: Example reasoning trace generated by grounded FULLPLAN planners. Invalid action plans
396 are manually inserted into the trace history and highlighted in blue. The correct continuations that
397 identify and correct these errors are shown in green, while incorrect continuations are shown in red.
398 The bottom figure visualizes the collision between two movements. RL planner better detects errors.
399
400
401

402 Table 3 reports the performance of our grounded planners on three unseen environments. We high-
403 light that the RL-trained planners consistently outperform SFT ones while maintaining plan effi-
404 ciency. For example, Qwen3-4B-RL FULLPLAN planner achieves 0.87 success rate on NEWCO-
405 ORD, 0.44 better than the SFT variant, showing better generalization of the reasoning capability.
406 This observation also aligns with previous RL for LLM works (Chu et al., 2025; Razin et al., 2025).
407
408
409

410 **Reasoning behavior change after RL.** Previous results have shown that RL training significantly
411 improves the planner’s planning ability. In this section, we analyze in more detail how the reasoning
412 behavior of the LLM-based planners changes before and after RL fine-tuning. Given the critical role
413 of reachability checks and collision checks in generating successful action plans, we prompt GPT-4o
414 to count the number of these checks in the reasoning traces produced by our FULLPLAN planners
415 across three BoxNet2D environment variants. The prompts used are provided in Appendix F.
416

417 As shown in Table 4, the RL planners per-
418 form more reachability checks (Rea.) and col-
419 lision checks (Col.) than the initial SFT planner.
420 These checks help ensure the feasibility of ac-
421 tion plans and lead to a large improvement in
422 success rate. This observation suggests that RL
423 training helps the LLM better understand the
424 importance of these checks and use them more
425 consistently.

426 We also verify their reasoning ability to identify and recover from intermediate errors by manually
427 injecting error steps into the trace. Specifically, we insert an invalid action that would lead to a
428 collision into the intermediate reasoning steps. To test whether the planner can recognize and correct
429 such errors, we append the phrase “Collision Check” to the perturbed trace to trigger verification.
430 Figure 4 shows two qualitative examples, where the injected invalid actions are highlighted in blue.
431 The LLM’s continuation is marked in green if it identifies and corrects the error, and in red if it
fails. We find that the RL-trained planner successfully finds the error and traces the issue to same
target position and overlapped movement paths. This suggests that RL helps build better physical
constraints-aware reasoning. Further quantitative analysis results are presented in Appendix C.2.

432 4.3 ABLATION STUDY
433

434 In this section, we explore the design choices for our framework on the BoxNet2D environment,
435 focusing on: ① How does SFT warmup affect final planner performance? ② Is the textual thinking
436 necessary for planners? ③ How does the efficiency penalty affect the planner’s behavior?

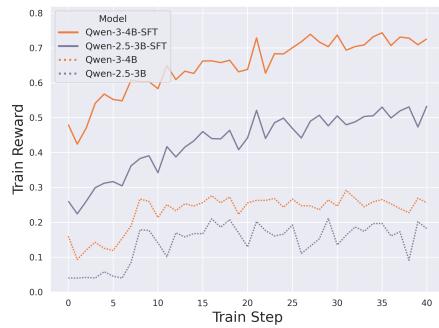
437 **Initial LLM policy matters.** Figure 5 shows how the training reward evolves over the first 40
438 steps for different initial FULLPLAN planners. For the original Qwen2.5-3 B-Instruct and Qwen3-4B
439 models, we observe a sharp reward increase of about 0.1 (the format reward) within the first 10 steps,
440 which indicates that they quickly learn to produce answers in required format. This observation
441 aligns with previous RLVR works (DeepSeek-AI, 2025). However, after this initial gain, the reward
442 plateaus, indicating limited additional learning to further improve the LLM’s planning capability.
443 In contrast, the LLM with SFT warmup shows a consistent increasing reward trend. This suggests that
444 the SFT warmup helps build a strong foundation, allowing it to continue learning and optimize effec-
445 tively in RL.
446

447 **Textual reasoning improves planning performance.** To assess the role of textual reasoning for
448 LLM planners, we perform an ablation study with our SFT and RL pipeline on Qwen-2.5-3B-Instruct.
449 In this experiment, we train a planner without the synthesized reasoning, *i.e.*, it generates only the final
450 action plan with no textual thinking. As shown in Table 5, removing intermediate reasoning leads to
451 a notable performance drop: success rates fall from 0.34 to 0.26 for SFT planner, and from 0.58 to 0.39
452 for RL planner. This shows the importance of textual thinking for LLM planners.
453

454 **Efficiency penalty in reward improves plan efficiency.** We observed a surprising finding that
455 RL-trained LLM planners produce more efficient plans than those generated by our hand-crafted A*
456 search algorithm in Table 2, which is likely due to the efficiency penalty term in the reward function.
457 To further understand its role, we conduct an ablation study on $r_{\text{efficiency}}$. Starting from the
458 same initial LLM policy Qwen-2.5-3B-SFT, we perform RL training without the efficiency penalty for
459 FULLPLAN planner. Table 5 presents the results. While both RL-finetuned LLM largely improve the success
460 rate, their plan efficiency differs significantly. The planner trained with $r_{\text{efficiency}}$ produces plans that are 2.09 steps
461 shorter. In contrast, the parallelism drops close to 1 without the penalty in RL. These results underscore the
462 importance of efficiency penalty in reward.
463

464 5 RELATED WORK
465

466 **Robotic planning and control with LLMs.** Classical robot task planning involves formalizing
467 goals and physical constraints with Temporal Logic or PDDL and solving them with constraint
468 solvers (Fox & Long, 2003; Emerson, 1990). LLM-based alternatives either select among motion
469 primitives (Guan et al., 2023; Skreta et al., 2023; Loula et al., 2025), generate code as a control
470 interface (Chen et al., 2025; Huang et al., 2022; Meng et al., 2025; Liang et al., 2023; Ahn et al.,
471 2022; Singh et al., 2023), combine with classical planners (Chen et al., 2024a; Lin et al., 2023), or
472 use multi-LLM discussion (Chen et al., 2024b; Zhang et al., 2024; Guo et al., 2024; Shen et al.,
473 2025b). Despite promising progress, many methods overly simplify physical constraints; we show
474 that even SOTA LLMs struggle under realistic constraints and introduce a method that grounds
475 smaller LLMs with constraint knowledge to substantially improve performance.
476



477 Figure 5: Training reward trajectory in first
478 40 steps for different initial LLMs.
479

480 Table 5: Impact of ablating thinking and
481 $r_{\text{efficiency}}$ on BoxNet2D performance.
482

Model	BoxNet2D		
	Success \uparrow	StepDiff. \downarrow	Para. \uparrow
Qwen2.5-SFT	0.34	0.11	1.51
– thinking	0.26	0.05	1.35
Qwen2.5-RL	0.58	-0.65	1.53
– thinking	0.39	-0.07	1.43
– $r_{\text{efficiency}}$	0.52	1.44	1.07

486 **Reinforcement learning with verifiable rewards for LLM reasoning.** RL with verifiable
 487 rewards has strengthened LLM reasoning in mathematics (Ren et al., 2025; Guo et al., 2025; Zeng
 488 et al., 2025), code (Liu & Zhang, 2025; Luo et al.; OpenAI, 2025), and multi-agent settings (Jin
 489 et al., 2025; Singh et al., 2025; Feng et al., 2024; OpenAI, 2024). Such training often induces emer-
 490 gent behaviors like feasibility checks and self-reflection, beyond supervised fine-tuning (Chu et al.,
 491 2025; Zelikman et al., 2024; Pan et al., 2025; Shen et al., 2025a; Hou et al., 2025). We extend RLVR
 492 to robotic control by integrating physical verification signals (collision avoidance, reachability, goal
 493 satisfaction) into training, thereby grounding constraint knowledge and yielding more robust plans.
 494 Additional related work discussion is provided in Appendix B.

495 6 CONCLUSION

496 In this paper, we present a novel framework that grounds LLMs with physical constraint knowl-
 497 edge, such as robot arm reachability and collision avoidance. By incorporating these constraints,
 500 LLMs are able to reason more effectively about action feasibility and generate efficient and physi-
 501 cally viable action plans. To evaluate our approach, we developed two BoxNet-based multi-robot
 502 environments, BoxNet2D and BoxNet 3D, both equipped with action feasibility checks. Experi-
 503 ments show that even small-scale LLMs at 3B and 4B parameter size, when grounded with constraint
 504 knowledge, significantly outperform larger SOTA LLMs without constraint knowledge training. Ad-
 505 dditional experiments on reasoning behavior and generalization further confirm that our models learn
 506 constraint-aware reasoning rather than simply overfitting to training data.

507 **Ethic Statement** This work aims to enhance the LLM-based multi-robot controller with an en-
 508 hanced training pipeline to incorporate physical constraint knowledge, such as reachability aware-
 509 ness and collision awareness, into the LLM planner. While our method performs better than off-the-
 510 shelf LLMs, it cannot achieve perfect accuracy under the considered multi-robot control setting. The
 511 effectiveness of our planner on more complex tasks, such as robot hand manipulation, is unexplored.
 512 Therefore, users should remain cautious when employing our planner in real-life deployment. We
 513 advise that users enforce additional verification and human overseeing when employing this the
 514 proposed pipeline in high-stake real-life robot control environments.

515 **Reproducibility Statement** Our work can be easily reproduced. We have included a detailed al-
 516 gorithm description in Section 2, and experiment setup in Section 4. We also include a detailed
 517 description of the environment implementation in Appendix D.3, the search algorithm we employed
 518 in Appendix D.4, and other implementation details in Appendix D. We also provide the code imple-
 519 mentation with corresponding data in the supplementary materials.

521 REFERENCES

523 Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea
 524 Finn, K. Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian
 525 Ibarz, Brian Ichter, A. Irpan, Eric Jang, Rosario M Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth,
 526 N. Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, S. Levine, Yao
 527 Lu, Linda Luu, Carolina Parada, P. Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse,
 528 D. Reyes, P. Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke,
 529 F. Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding
 530 language in robotic affordances. *Conference on Robot Learning*, 2022.

531 Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. Au-
 532 totamp: Autoregressive task and motion planning with llms as translators and checkers. In *2024*
 533 *IEEE International conference on robotics and automation (ICRA)*, pp. 6695–6702. IEEE, 2024a.

534 Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multi-robot
 535 collaboration with large language models: Centralized or decentralized systems? In *2024 IEEE*
 536 *International Conference on Robotics and Automation (ICRA)*, pp. 4311–4317. IEEE, 2024b.

538 Yongchao Chen, Yilun Hao, Yang Zhang, and Chuchu Fan. Code-as-symbolic-planner: Foundation
 539 model-based robot planning via symbolic code generation. *arXiv preprint arXiv: 2503.01700*,
 2025.

540 Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V.
 541 Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation
 542 model post-training. *arXiv preprint arXiv: 2501.17161*, 2025.

543

544 DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.
 545 *arXiv preprint arXiv: 2501.12948*, 2025.

546

547 E Allen Emerson. Temporal and modal logic. In *Formal models and semantics*, pp. 995–1072.
 548 Elsevier, 1990.

549

550 Peiyuan Feng, Yichen He, Guanhua Huang, Yuan Lin, Hanchong Zhang, Yuchen Zhang, and Hang
 551 Li. Agile: A novel reinforcement learning framework of llm agents. *Neural Information Processing-
 552 ing Systems*, 2024.

553

554 Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning
 555 domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

556

557 Lin Guan, Karthik Valmeekam, Sarah Sreedharan, and Subbarao Kambhampati. Leveraging pre-
 558 trained large language models to construct and utilize world models for model-based task plan-
 559 ning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023.

560

561 Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1: Incentivizing reasoning capability in
 562 llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

563

564 Xudong Guo, Kaixuan Huang, Jiale Liu, Wenhui Fan, Natalia Vélez, Qingyun Wu, Huazheng Wang,
 565 Thomas L Griffiths, and Mengdi Wang. Embodied llm agents learn to cooperate in organized
 566 teams. *arXiv preprint arXiv:2403.12482*, 2024.

567

568 Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh
 569 Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*,
 570 2024.

571

572 Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang.
 573 Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. 2025. URL
 574 <https://openreview.net/forum?id=gahaeltNo>.

575

576 Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan
 577 Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through
 578 planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

579

580 Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1:
 581 Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint
 582 arXiv: 2503.09516*, 2025.

583

584 Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine.
 585 Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language
 586 grounding. *arXiv preprint arXiv: 2501.04693*, 2025.

587

588 Lucas Lehnert, Sainbayar Sukhbaatar, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond
 589 a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:
 590 2402.14083*, 2024.

591

592 Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and
 593 Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE
 594 International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

595

596 Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2motion:
 597 From natural language instructions to feasible plans. *arXiv preprint arXiv: 2303.12153*, 2023.

598

599 Jiawei Liu and Lingming Zhang. Code-r1: Reproducing r1 for code with reliable rewards. 2025.

600

601 I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *International Conference on
 602 Learning Representations*, 2017.

594 João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu,
 595 Yahya Emara, Marjorie Freedman, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Alexan-
 596 der K. Lew, Tim Vieira, and Timothy J. O'Donnell. Syntactic and semantic control of large lan-
 597 guage models via sequential monte carlo. In *The Thirteenth International Conference on Learn-
 598 ing Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL
 599 <https://openreview.net/forum?id=xoXn62FzD0>.

600 Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi,
 601 Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica.
 602 Deepcoder: A fully open-source 14b coder at o3-mini level.
 603

604 Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large
 605 language models, 2023.

606 Yue Meng, Fei Chen, Yongchao Chen, and Chuchu Fan. Audere: Automated strategy decision and
 607 realization in robot planning and control via llms. *arXiv preprint arXiv: 2504.03015*, 2025.

608 OpenAI. Introducing operator, 2024. URL <https://openai.com/index/introducing-operator/>. Accessed: 2025-03-23.

609 OpenAI. Competitive programming with large reasoning models. *arXiv preprint arXiv: 2502.06807*,
 610 2025.

611 Jiazheng Pan, Che Liu, Junde Wu, Fenglin Liu, Jiayuan Zhu, Hongwei Bran Li, Chen Chen, Cheng
 612 Ouyang, and Daniel Rueckert. Medvilm-r1: Incentivizing medical reasoning capability of vision-
 613 language models (vlms) via reinforcement learning. *arXiv preprint arXiv:2502.19634*, 2025.

614 Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D. Lee, and Sanjeev Arora. What
 615 makes a reward model a good teacher? an optimization perspective. *arXiv preprint arXiv: 2503.15477*, 2025.

616 Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang,
 617 Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan
 618 Liu, Wenjun Gao, Daya Guo, and Chong Ruan. Deepseek-prover-v2: Advancing formal math-
 619 ematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv: 2504.21801*, 2025.

620 John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-
 621 dimensional continuous control using generalized advantage estimation. *arXiv preprint
 622 arXiv:1506.02438*, 2015.

623 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
 624 optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

625 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 626 Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathe-
 627 matical reasoning in open language models. *arXiv preprint arXiv: 2402.03300*, 2024.

628 Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun
 629 Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large
 630 vision-language model. *arXiv preprint arXiv:2504.07615*, 2025a.

631 Zhixuan Shen, Haonan Luo, Kexun Chen, Fengmao Lv, and Tianrui Li. Enhancing multi-robot
 632 semantic navigation through multimodal chain-of-thought score collaboration. In *Proceedings of
 633 the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14664–14672, 2025b.

634 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,
 635 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint
 636 arXiv: 2409.19256*, 2024.

637 Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter
 638 Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using
 639 large language models. In *2023 IEEE International Conference on Robotics and Automation
 640 (ICRA)*, pp. 11523–11530. IEEE, 2023.

648 Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool
 649 integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.
 650

651 Marta Skreta, Naruki Yoshikawa, Sebastian Arellano-Rubach, Zhi Ji, Lasse Bjørn Kristensen,
 652 Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. Errors are useful
 653 prompts: Instruction guided task programming with verifier-assisted iterative prompting. *arXiv
 654 preprint arXiv:2303.14100*, 2023.

655

656 DiJia Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. Dualformer:
 657 Controllable fast and slow thinking by learning with randomized reasoning traces. In *The Thir-
 658 teenth International Conference on Learning Representations*, 2024.

659

660 Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuchu Fan. Multi-agent motion planning from signal
 661 temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(2):3451–3458, 2022.

662

663 Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control.
 664 pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

665

666 Eric Zelikman, Georges Raif Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah Goodman.
 667 Quiet-star: Language models can teach themselves to think before speaking. In *First Conference
 668 on Language Modeling*, 2024.

669

670 Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-
 671 zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv
 672 preprint arXiv: 2503.18892*, 2025.

673

674 Songyuan Zhang, Oswin So, Kunal Garg, and Chuchu Fan. Gcbf+: A neural graph control barrier
 675 function framework for distributed safe multi-agent control. *IEEE Transactions on Robotics*,
 2025.

676

677 Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. Towards
 678 efficient llm grounding for embodied multi-agent collaboration. *arXiv preprint arXiv:2405.14314*,
 2024.

679

680 A LIMITATIONS AND SOCIETAL IMPACTS

681 Our work introduces a novel framework to ground LLMs with physical constraint knowledge for
 682 robot control tasks, significantly enhancing their ability to reason about action feasibility during
 683 plan generation. This leads to substantial improvements in planning performance. We validate the
 684 effectiveness of our approach through experiments in two BoxNet-based multi-robot environments
 685 on two small-scale LLMs. However, this work has two limitations: ① Our experiments are lim-
 686 ited to the BoxNet task due to the high implementation overhead required for other robot control
 687 environments. Extending our framework to additional physical constraint-sensitive tasks remains
 688 an important direction for future work. ② The RL training is conducted at a limited scale due to
 689 computational constraints. Unlike typical RL setups in math and coding domains that allow for
 690 training over multiple epochs (Liu & Zhang, 2025; Luo et al.), our training is restricted to just one or
 691 two epochs. Despite this limitation, our experimental results already demonstrate strong reasoning
 692 capabilities on robotic tasks.

693 Our work aims to advance the integration of LLMs into robotic control planning, which has many
 694 promising societal benefits. By enabling LLMs to better understand and operate within physical
 695 constraints, LLMs can help build safer, more reliable, and more efficient multi-robot systems. This
 696 can potentially enhance robotic automation domains heavily involving many robots. In particular,
 697 improved planning performance can reduce operational errors and increase productivity. However,
 698 as with any deployment of AI in real-world decision-making systems, there are potential risks if
 699 the planners are deployed with a dangerous purpose. Future extensions of this work should also
 700 consider robustness to adversarial scenarios to ensure responsible real-world integration.

702 **B ADDITIONAL RELATED WORK**
703704 **B.1 ROBOTIC PLANNING AND CONTROL WITH LLMs**
705

706 Robotic planning and control is a complex task that requires high-level planning under various phys-
707 ical constraints. Traditional approaches typically translate task goals and physical constraints into
708 formal logic specifications, such as Temporal Logic or PDDL (Fox & Long, 2003; Emerson, 1990),
709 and solve them using constraint solvers. More recently, LLMs have been applied to robotic control
710 due to their strong reasoning capabilities and better scalability compared to constraint solvers. For
711 example, some works use LLMs to choose actions from predefined motion primitives (Guan et al.,
712 2023; Skreta et al., 2023; Loula et al., 2025), while others treat code as an intermediate representa-
713 tion for control (Chen et al., 2025; Huang et al., 2022; Meng et al., 2025; Liang et al., 2023; Ahn
714 et al., 2022; Singh et al., 2023). Hybrid approaches, such as AutoTAMP (Chen et al., 2024a) and
715 Text2Motion (Lin et al., 2023), combine traditional planning tools with LLMs for action planning.
716 Another series of works employs multi-LLM discussion for robotic tasks (Chen et al., 2024b; Mandi
717 et al., 2023; Zhang et al., 2024; Guo et al., 2024; Shen et al., 2025b). While these methods show
718 promising results, many of them overly simplify physical constraints, limiting their real-world ap-
719 plicability. In this work, we demonstrate that even SOTA LLMs struggle under realistic physical
720 constraints, and further introduce a novel approach that grounds smaller LLMs with this constraint
721 knowledge, which largely improves performance.

722 **B.2 REINFORCEMENT LEARNING WITH VERIFIABLE REWARDS FOR LLM REASONING**
723

724 Reinforcement learning (RL) has demonstrated significant promise in enhancing the reasoning ca-
725 pabilities of large language models (LLMs) across a wide range of domains, including mathemat-
726 ics (Ren et al., 2025; Guo et al., 2025; Zeng et al., 2025), code generation (Liu & Zhang, 2025;
727 Luo et al.; OpenAI, 2025), and complex multi-agent systems (Jin et al., 2025; Singh et al., 2025;
728 Feng et al., 2024; OpenAI, 2024). A common paradigm involves training LLMs to optimize for a
729 verifiable reward, such as the correctness of a math solution or whether the generated code passes
730 unit tests, using RL training. Many previous works show that the RL training process vastly im-
731 proves LLM reasoning (Chu et al., 2025; Feng et al., 2024; Pan et al., 2025; Shen et al., 2025a;
732 Hou et al., 2025). The improvement is often accompanied by emergent reasoning behaviors such
733 as feasibility checks and self-reflection, which are difficult to elicit through supervised fine-tuning
734 alone (Chu et al., 2025; Zelikman et al., 2024; Hosseini et al., 2024). In this work, we extend the
735 RLVR to robotic control, with a focus on grounding LLMs with knowledge of physical constraints.
736 Our method leverages RLVR to teach LLMs to reason under the constraints inherent in robotic
737 planning tasks, such as collision avoidance, reachability accordance, and goal satisfaction. By in-
738 tegrating physical verification signals into the training process, the model learns to internalize these
739 constraints as part of its reasoning process. This grounding leads to more robust and reliable control
740 planning decisions in downstream robotic applications.

741 **C ADDITIONAL EXPERIMENTAL RESULTS AND ANALYSIS**
742743 **C.1 COMPARISON WITH SYMBOLIC LANGUAGE TRANSLATION-BASED METHODS**
744

745 While our main paper focuses on prompting an LLM to generate robot control plans, there is a
746 series of work employing LLM to generate intermediate representations of the task, such as the tem-
747 poral logic language (Chen et al., 2024a; Emerson, 1990). To ensure a comprehensive comparison
748 between our method and previous work, we follow the AutoTAMP setup and prompt two strong
749 LLMs, GPT-4o and GPT-4o-mini, to generate Python function calls to our A* search algorithm for
750 solving the given task, *i.e.*, translate the given task to Python language and utilize a solver engine to
751 complete the task, similar to previous PDDL solver methods (Fox & Long, 2003; Chen et al., 2025).

752 Specifically, we evaluate two translation schemes on BoxNet2D: *input*, where the LLM converts
753 the environment textual description into two lists representing initial object and robot arm positions;
754 and *code*, where the LLM generates a full Python function call, including argument construction
755 and invocation, e.g., `env = BoxNet2D(); a_star_search(env)`. We note that *input* is a simplified task,
756 as it only requires the LLM to summarize the environment description without performing full code
757 synthesis.

Table 6 below summarizes the results. We highlight that directly combining an off-shelf LLM for AutoTAMP style translation performs worse than our fine-tuned planner, potentially due to the *inability to understand the translation* process, even for the simplified input synthesis. For a more complicated full-function call translation, the performance is even worse. On the other hand, we note that our SFT-then-RL pipeline can also be combined with AutoTAMP-style translation, where LLM is fine-tuned to generate correct translation and achieve high performance. We leave the exploration for this control scheme in future work.

Table 6: Success rate of AutoTamp baseline and our planner on BoxNet2D environment.

Model	Success ↑
AutoTamp (Input)	
GPT-4omini	0.60
GPT-4o	0.72
AutoTamp (Code)	
GPT-4omini	0.38
GPT-4o	0.45
Our FULLPLAN planner	
Qwen2.5-3B-RL	0.64
Qwen3-4B-RL	0.76

C.2 ABILITY TO IDENTIFY AND RECOVER FROM INTERMEDIATE ERRORS

Figure 4 presents an example of LLM planner identifying and recovering from the manually inserted invalid actions. We further extend the evaluation to a quantitative analysis in this section. Specifically, we collect 50 correct reasoning traces on BoxNet2D environment and manually insert an invalid action step in using the fixed template “Robot x can move Object y to coordinate”. Here, we create an incorrect action by setting an unreachable object for a robot or setting the coordinate to a collision point, which are denoted with *Unreachable* and *Collision*, respectively. Then, a FULLPLAN planner continues generation at the end of this invalid action sentence. The table below reports the final task success rate after this perturbation. We note that the performance remains stable, with a slight drop of around 3% accuracy. This further validates the observation from our qualitative analysis.

Table 7: Success rate on BoxNet2D environment under different reasoning step perturbations.

Model	Original ↑	Unreachable ↑	Collision ↑
FULLPLAN Success Rate			
Qwen2.5-3B-SFT	0.44	0.42	0.40
Qwen2.5-3B-RL	0.64	0.62	0.60
Qwen3-4B-SFT	0.58	0.52	0.54
Qwen3-4B-RL	0.76	0.72	0.70

C.3 PLANNER ERROR TYPE BREAKDOWN

To better understand the reasoning errors in our fine-tuned LLM planners, we perform a detailed breakdown of failure types. Specifically, we categorize four different outcome types for the generated plan: ① *Success*, where the planner successfully completes the task with all objects correctly moved to their target positions; ② *Unreachable Position*, where the planner attempts to move an object to a location that is not reachable for a certain robot; ③ *Collision*, where the generated plan results in object or robot collisions during execution; ④ *Incomplete execution*, where the planner fails to move all required objects to their target positions by the end of execution despite no invalid action. Table 8 presents the full results for our fine-tuned planner on BoxNet2D environment.

We note that the most invalid plans are from *unreachable position* and *collision* for the FULLPLAN planner, which is likely due to the difficulty in predicting all intermediate object positions during execution. On the other hand, collision and incomplete execution are two main sources for the failures of the REPLAN planner.

Table 8: Error type breakdown for finetuned LLM planners on BoxNet2D environment. Each number is the ratio of outcome types among all tested environments.

Model	Success \uparrow	Unreachable Pos. \downarrow	Collision \downarrow	Incomplete Exec. \downarrow
FULLPLAN Breakdown				
Qwen2.5-3B-SFT	0.34	0.37	0.24	0.05
Qwen2.5-3B-RL	0.58	0.23	0.15	0.04
Qwen3-4B-SFT	0.45	0.36	0.17	0.02
Qwen3-4B-RL	0.87	0.06	0.05	0.02
REPLAN Breakdown				
Qwen2.5-3B-SFT	0.30	0.18	0.30	0.22
Qwen2.5-3B-RL	0.68	0.02	0.11	0.19
Qwen3-4B-SFT	0.31	0.11	0.31	0.27
Qwen3-4B-RL	0.75	0.03	0.07	0.15

C.4 UNSOLVABLE TASK IN ENVIRONMENT

In this work, we mainly consider solvable tasks for both BoxNet2D and BoxNet3D. However, there is a potential scenario where the given task is unsolvable. While our work is not designed to handle this case, we show that further fine-tuning of our LLM planner on a combination of solvable task environments and unsolvable task environments can inject this ability to the planner in this section. Specifically, we follow the original SFT-and-RL training pipeline and further fine-tune the BoxNet2D FULLPLAN planner by adding 1,000 unsolvable BoxNet2D environments to the training set. During training, we instruct the planner to respond with a refusal sentence “This is an unsolvable environment” and avoid generating any action plan.

Table 9 presents the results, where we evaluate the updated planner (denoted by * suffix) on the original test set and 50 unseen unsolvable environments. As shown in the table, the updated Qwne3-4B-RL* planner achieves 98% detection accuracy on unsolvable cases while maintaining original performance on solvable environments, indicating that we can easily adapt the framework to incorporate unsolvable task awareness to the planner.

Table 9: Success rate and unsolvable case detection rate for RL-tuned LLM planners on BoxNet2D environment.

Model	Success \uparrow	Unsolvable Detection \uparrow
FULLPLAN Planner		
Qwen2.5-3B-RL	0.34	0.00
Qwen3-4B-RL	0.58	0.01
Qwen2.5-3B-RL*	0.45	0.96
Qwen3-4B-RL*	0.87	0.98

D ADDITIONAL IMPLEMENTATION DETAILS

In this section, we provide more implementation details, including: the RL training algorithm (Section D.1), the implementation of BoxNet2D, BoxNet3D and dataset statistics (Section D.3), training efficiency analysis(Section D.2) and the A* search algorithm for data generation (Section D.4).

D.1 GRPO ALGORITHM

GRPO (Shao et al., 2024), or group relative policy optimization, is a variant of PPO algorithm (Schulman et al., 2017) proposed for LLM RL. In this section, we briefly outline the GRPO algorithm and refer readers to the original paper (DeepSeek-AI, 2025) for more details.

As we mentioned in the main paper, given an initial LLM policy \mathcal{M}_{θ_0} and a train dataset \mathcal{D} , the GRPO loss is defined as follows:

$$\begin{aligned} \mathcal{J}_{GRPO}(\mathcal{M}_{\theta_i}) &= \mathbb{E} \left[\left(\mathbf{q} \sim \mathcal{D}, \{\mathbf{s}_j\}_{j=1}^G \sim \mathcal{M}_{\theta_{i-1}}(\mathbf{O}|\mathbf{q}; \mathcal{C}) \right) \right] \\ &= \frac{1}{G} \sum_{j=1}^G \left(\min \left(\frac{\mathcal{M}_{\theta}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})}{\mathcal{M}_{\theta_{i-1}}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})} \right) A_j, \text{clip} \left(\frac{\mathcal{M}_{\theta}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})}{\mathcal{M}_{\theta_{i-1}}(\mathbf{s}_j|\mathbf{q}; \mathcal{C})}, 1 - \epsilon, 1 + \epsilon \right) A_j \right) \\ &\quad - \beta \mathbb{D}_{KL}(\mathcal{M}_{\theta_i} \parallel \mathcal{M}_{\theta_0}), \end{aligned}$$

where i denotes the train step, G denotes the group size, \mathbf{q} denotes a textual query describing a robotic control task, \mathcal{C} denotes the constraints in text, A_j denotes the advantage for j -th rollout s_j . The definition of advantage is:

$$A_j = \frac{r_j - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})},$$

given the reward $\mathbf{r} = \{r_1, \dots, r_G\}$ for all LLM rollouts to the query task \mathbf{q} , following the Generalized Advantage Estimation (GAE) (Schulman et al., 2015).

D.2 TRAINING EFFICIENCY ANALYSIS

In this section, we include detailed statistics on the training cost of our method in Table 10. **In general, our training is relatively light-weight since the base LLM is small in scale.** We note that the GPU hours are much larger for RL training due to the heavy cost in generating a batch of new responses, given the same input for GRPO training.

Table 10: Computation cost and data usage for BoxNet2D and BoxNet3D experiments. We employ H100 as the GPU.

Model	Step	GPU Hour	Data num
BoxNet2D			
Qwen2.5-3B-SFT	4296	4.3	220,000
Qwen2.5-3B-RL	160	87.2	5,120
Qwen3-4B-SFT	4296	5.5	220,000
Qwen3-4B-RL	160	102.5	5,120
BoxNet3D			
Qwen2.5-3B-SFT	562	1.9	9,000
Qwen2.5-3B-RL	120	184.8	3,840
Qwen3-4B-SFT	562	2.25	9,000
Qwen3-4B-RL	120	194.4	3,840

D.3 BOXNET ENVIRONMENT IMPLEMENTATION AND STATISTICS

We implement two different environments based on the BoxNet task, which involves multiple robots in a grid map and collaborating to move objects to the corresponding target positions. Both environments are implemented in Python.

BoxNet2D For BoxNet2D, we manually implement the feasibility check by calculating the relative geometric position of robot arms and objects. The map size ranges from 2×2 to 6×6 , and the object number ranges from 1 to 5. With the default robot placement strategy, the number of robots involved ranges from 1 to 25 for different grid size configurations. For each unique map configuration, *i.e.*, a tuple of map width, height, and the object number, we randomly generate at most 150 different object initial and target positions to construct the unique environments in train dataset. The testing data consists of the square maps with the width ranging from 2 to 6, and the object number ranges from 1 to 5. We generate at most 10 unique environments to construct the test dataset. The detailed dataset statistics are summarized in Table 11.

In the unseen environment transfer experiment, we generate two variants of BoxNet2D test set: RANDROB, where the robot position is not evenly placed at the grid joints, and NEWCOORD, where the object position coordinates are perturbed with a random offset. For RANDROB, all robots are placed in a connected manner, meaning that all objects can be reached by a robot.

Table 11: Dataset statistics of BoxNet2D and BoxNet3D. The average steps to complete and parallelism are all based on the optimal plans generated by our manually implemented A* algorithm.

Dataset	Sample No.	Avg. Step	Para.	Avg. Robot
BoxNet2D-train	55000	8.13	1.73	6.25
BoxNet2D-test	250	8.32	1.75	6.25
RANDROB	200	7.06	1.49	6.25
NEWCOORD	250	8.59	1.77	6.25
BoxNet3D-train	1800	6.27	1.89	2.25
BoxNet3D-test	160	5.62	1.88	2.25

BoxNet3D For BoxNet3D, we use MuJoCo to implement the feasibility checks such as robot arm collision and object collisions. The map size ranges from 2×2 to 4×4 , and the object number ranges from 1 to 4. With the default robot placement strategy, the number of robots involved ranges from 1 to 9 for different grid size configurations. For each map configuration, we randomly generate at most 100 different environments for the training data and at most 5 for the test data. Detailed dataset statistics are summarized in Table 11

D.4 A* SEARCH ALGORITHM

We implement an A* search algorithm for solving the generated task. At each search step, the general workflow is: ① select the current best environment state, ② generate valid action for a single robot, ③ combine multiple valid actions and check whether they can run in parallel, and ④ update the environment with potential next step actions and put to candidate tools for next search step.

We list a Python reference code below:

Listing 1: Reference A* search implementation

```
def astar_search(env: Any, max_iterations: int = 1000):
    open_set = []
    closed_set = set()

    g_scores: Dict[int, float] = {}
    came_from: Dict[int, Tuple[Optional[int], Optional[str]]] = {}
    states_cache: Dict[int, EnvStates] = {}

    try:
        initial_state_data = env.get_states()
        current_state = EnvStates(env, current_state_data=initial_state_data)
        initial_hash = current_state.hash()
    except Exception as e:
        raise e

    g_scores[initial_hash] = 0.0
    came_from[initial_hash] = (None, None)
    states_cache[initial_hash] = current_state

    heapq.heappush(open_set, (current_state.heuristic(), random.random(),
                             initial_hash))

    iterations = 0

    while open_set and iterations < max_iterations:
        iterations += 1

        f_val, _, current_hash = heapq.heappop(open_set)

        if current_hash in closed_set:
            continue

        current_state = states_cache[current_hash]
        closed_set.add(current_hash)

        if current_state.is_goal():
            return reconstruct_path(came_from, current_hash)

        for action, next_state in current_state.transitions.items():
            next_hash = next_state.hash()
            if next_hash in closed_set:
                continue

            g_score = g_scores.get(current_hash, 0.0) + current_state.heuristic(next_state)
            if next_hash not in g_scores or g_score < g_scores[next_hash]:
                g_scores[next_hash] = g_score
                came_from[next_hash] = (current_hash, action)
                states_cache[next_hash] = next_state

                heapq.heappush(open_set, (g_score, random.random(), next_hash))
```

```

972
973
974     potential_next_moves = generate_potential_actions(current_state)
975
976     for action_str, next_state_obj in potential_next_moves:
977         if next_state_obj is None:
978             continue
979
980         next_hash = next_state_obj.hash()
981         if next_hash in closed_set:
982             continue
983
984         cost_of_this_action = 1.0
985         tentative_g_score = g_scores[current_hash] + cost_of_this_action
986
987         if tentative_g_score < g_scores.get(next_hash, float('inf')):
988             came_from[next_hash] = (current_hash, action_str)
989             g_scores[next_hash] = tentative_g_score
990             states_cache[next_hash] = next_state_obj
991
992             f_score_neighbor = tentative_g_score + next_state_obj.heuristic()
993             heappush(open_set, (f_score_neighbor, random.random(),
994                                 next_hash))
995
996     return None
997
998 class EnvStates:
999     _hash_val: Optional[int] = None
1000
1001     def __init__(self, env: Any, parent_state_data=None, current_state_data=None):
1002         self.env = env
1003         self.parent_state_data = parent_state_data
1004         self.cur_states = current_state_data
1005         # Assumes env has get_target_pos() and can define retraction height
1006         # internally or via config
1007         self.target_positions = self.env.get_target_pos() # Target positions should
1008         # include Z if relevant
1009
1010     def hash(self) -> int:
1011         if self._hash_val is not None:
1012             return self._hash_val
1013         self._hash_val = xxhash.xxh64(self.cur_states.tobytes()).intdigest()
1014         return self._hash_val
1015
1016     def _box_positions(self):
1017         self.env.reset(states=self.cur_states)
1018         return [
1019             boxname: self.env.get_box_pos(boxname)
1020             for boxname in self.env.object_names
1021         ]
1022
1023     def arm_positions(self):
1024         self.env.reset(states=self.cur_states)
1025         return [
1026             robot_name: self.env.get_arm_pos(robot_name)
1027             for robot_name in self.env.robot_names
1028         ]
1029
1030     def is_goal(self) -> bool:
1031         current_box_pos_map = self._box_positions()
1032         if len(current_box_pos_map) != len(self.target_positions):
1033             return False
1034
1035         for box_name, target_val in self.target_positions.items():
1036             if box_name not in current_box_pos_map:
1037                 return False
1038             # Use new env method for checking if object is at its target
1039             if not self.env.is_object_at_target(current_box_pos_map[box_name],
1040                                               target_val, box_name):
1041                 return False
1042
1043         return True
1044
1045     def heuristic(self) -> float:
1046         self.env.reset(states=self.cur_states)
1047         current_obj_positions_map = self._box_positions()
1048
1049         h = 0.0
1050         num_matched = 0
1051
1052         for name, target_pos_val in self.target_positions.items():
1053             if name in current_obj_positions_map:
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
313
```

```

1026
1027
1028     current_pos_val = current_obj_positions_map[name]
1029     if np.isnan(current_pos_val): # Check for NaN
1030         return float("inf")
1031     # Use env method for calculating distance/cost component for
1032     # heuristic
1033     h += self.env.calculate_placement_quality(current_pos_val,
1034     target_pos_val, name)
1035     num_matched +=1
1036 else: # Object in target not found in current state
1037     return float("inf")
1038
1039     if num_matched != len(self.target_positions): # Not all target objects were
1040     found or matched
1041         return float("inf")
1042
1043     return math.sqrt(h) if h > 0 else 0.0
1044
1045 def apply_actions(self, action_strings: Union[List[str], str]) ->
1046     Optional["EnvStates"]:
1047     self.env.reset(states=self.cur_states)
1048     action_input = action_strings
1049     if isinstance(action_strings, list):
1050         action_input = "\n".join(action_strings)
1051
1052     out = self.env.simulate_one_step(action_input)
1053     if out["success"]:
1054         return EnvStates(
1055             self.env,
1056             parent_state_data=self.cur_states,
1057             current_state_data=self.env.get_states(),
1058         )
1059     else:
1060         return None
1061
1062 # --- Utility Functions (Domain-specific helpers removed, env handles them) ---
1063
1064 def generate_single_robot_action(robot_id: str, state: EnvStates) -> List[str]:
1065     robot_actions = []
1066     env = state.env # Get the environment reference
1067     base_pos = env.get_base_pos(robot_id)
1068     arm_pos = state.arm_positions()[robot_id]
1069
1070     for obj_id, obj_pos_val in state._box_positions().items():
1071         target_pos_val = state.target_positions[obj_id]
1072
1073         if env.check_reach_range(robot_id, obj_pos_val): # Existing env call for
1074             # reachability
1075             if env.is_object_at_target(obj_pos_val, target_pos_val, obj_id):
1076                 continue
1077
1078             # Get potential next positions for the object from the environment
1079             potential_next_obj_placements = env.get_valid_next_object_positions(
1080                 obj_id, obj_pos_val, robot_id, base_pos
1081             )
1082             current_placement_quality = env.calculate_placement_quality(obj_pos_val,
1083             target_pos_val, obj_id)
1084
1085             action_candidates_for_obj = []
1086
1087             # Try to move the object to a better position
1088             for next_obj_p in potential_next_obj_placements:
1089                 if env.calculate_placement_quality(next_obj_p, target_pos_val,
1090                 obj_id) < current_placement_quality:
1091                     # Check if arm is already at the object
1092                     if not env.is_arm_at_position(arm_pos, obj_pos_val[:2],
1093                     robot_id):
1094                         # Action: Move arm to object
1095                         action_str = env.format_move_action_string(robot_id,
1096                         obj_pos_val[:2], False)
1097                         action_candidates_for_obj.append(action_str)
1098
1099                     # Action: Move object (arm is now assumed to be at object or
1100                     # will be moved by first action)
1101                     action_str = env.format_move_action_string(robot_id,
1102                         next_obj_p[:2], True)
1103                     action_candidates_for_obj.append(action_str)
1104
1105             # Try to move arm to an alternative/safe position if not productively
1106             # moving an object
1107             if not action_candidates_for_obj or all(
1108
1109

```

```

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

    # Check if any action involves carrying (True flag)
    # This logic might need refinement based on how
    # format_move_action_string works
    # or if we have a better way to check if an action is "productive"
    "True" not in act for act in action_candidates_for_obj
):
    alternative_arm_destinations = env.get_alternative_arm_destinations(
        robot_id, base_pos, arm_pos
    )
    if alternative_arm_destinations:
        # Environment can decide which one to pick or return just one
        chosen_alt_dest = alternative_arm_destinations[0] # Take the
        # first one
        if not env.is_arm_at_position(arm_pos, chosen_alt_dest[:2],
        robot_id):
            action_str = env.format_move_action_string(robot_id,
            chosen_alt_dest[:2], False)
            action_candidates_for_obj.append(action_str)

    for action_str_candidate in action_candidates_for_obj:
        if action_str_candidate not in robot_actions:
            robot_actions.append(action_str_candidate)

    return robot_actions

def verify_parallel_actions(actionstr_input: Union[List[str], str], state:
    EnvStates) -> Tuple[bool, Optional[EnvStates]]:
    current_env = state.env
    current_env.reset(states=state.cur_states)

    action_to_simulate = actionstr_input
    if isinstance(actionstr_input, list):
        action_to_simulate = "\n".join(actionstr_input)

    out = current_env.simulate_one_step(action_to_simulate)

    if out["success"]:
        new_state_data = current_env.get_states()
        new_search_state = EnvStates(
            current_env,
            parent_state_data=state.cur_states,
            current_state_data=new_state_data,
        )
        return True, new_search_state
    else:
        return False, None

def generate_potential_actions(state: EnvStates) -> List[Tuple[str, EnvStates]]:
    if not hasattr(state.env, 'robot_names'):
        return []
    robot_names = sorted(state.env.robot_names)

    single_robot_potential_actions: Dict[str, List[str]] = {
        r: generate_single_robot_action(r, state) for r in robot_names
    }

    valid_action_sets: List[Tuple[str, EnvStates]] = []
    action_verification_tasks = []

    for robot_id, actions in single_robot_potential_actions.items():
        for action in actions:
            action_verification_tasks.append(([action], state))

    active_robots = [r for r in robot_names if single_robot_potential_actions[r]]
    if len(active_robots) >= 2:
        max_concurrent_robots = 2
        if hasattr(state.env, 'object_names'): # Check if object_names exists before
        # using its length
            max_concurrent_robots = min(4, len(state.env.object_names),
            len(active_robots))
        else: # Fallback if object_names is not available
            max_concurrent_robots = min(2, len(active_robots))

    for group_size in range(2, max_concurrent_robots + 1):
        if group_size > len(active_robots): continue
        for robot_group in itertools.combinations(active_robots, group_size):

```

```

1134
1135         action_combos_for_group = list(
1136             itertools.product(
1137                 *[single_robot_potential_actions[r] for r in robot_group]
1138             )
1139         )
1140         cleaned_combos = [[a for a in combo] for combo in
1141             action_combos_for_group]
1142         for combo in cleaned_combos:
1143             action_verification_tasks.append((combo, state))
1144
1145     verify_results = []
1146     for action_combo, original_state_for_verification in action_verification_tasks:
1147         success, new_state_obj = verify_parallel_actions(action_combo,
1148             ↪ original_state_for_verification)
1149         if success and new_state_obj is not None:
1150             verify_results.append("\n".join(action_combo), new_state_obj)
1151
1152     verify_results.sort(
1153         key=lambda x_tuple: (x_tuple[1].heuristic(), -x_tuple[0].count("True"),
1154             ↪ -len(x_tuple[0].split("\n"))))
1155     )
1156
1157     return verify_results[:20]
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3097
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3137
3138
3139
3139
3140
3141
3142
3143
3143
3144
3145
3145
3146
3147
3147
3148
3149
3149
3150
3151
3151
3152
3152
3153
3153
3154
3154
3155
3155
3156
3156
3157
3157
3158
3158
3159
3159
3160
3160
3161
3161
3162
3162
3163
3163
3164
3164
3165
3165
3166
3166
3167
3167
3168
3168
3169
3169
3170
3170
3171
3171
3172
3172
3173
3173
3174
3174
3175
3175
3176
3176
3177
3177
3178
3178
3179
3179
3180
3180
3181
3181
3182
3182
3183
3183
3184
3184
3185
3185
3186
3186
3187
3187
3188
3188
3189
3189
3190
3190
3191
3191
3192
3192
3193
3193
3194
3194
3195
3195
3196
3196
3197
3197
3198
3198
3199
3199
3200
3200
3201
3201
3202
3202
3203
3203
3204
3204
3205
3205
3206
3206
3207
3207
3208
3208
3209
3209
3210
3210
3211
3211
3212
3212
3213
3213
3214
3214
3215
3215
3216
3216
3217
3217
3218
3218
3219
3219
3220
3220
3221
3221
3222
3222
3223
3223
3224
3224
3225
3225
3226
3226
3227
3227
3228
3228
3229
3229
3230
3230
3231
3231
3232
3232
3233
3233
3234
3234
3235
3235
3236
3236
3237
3237
3238
3238
3239
3239
3240
3240
3241
3241
3242
3242
3243
3243
3244
3244
3245
3245
3246
324
```

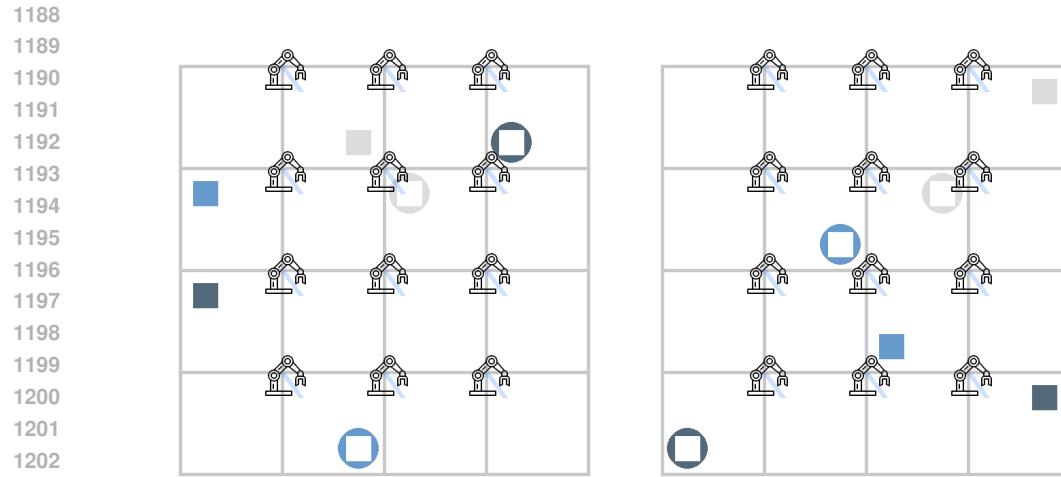


Figure 6: Example BoxNet2D environment

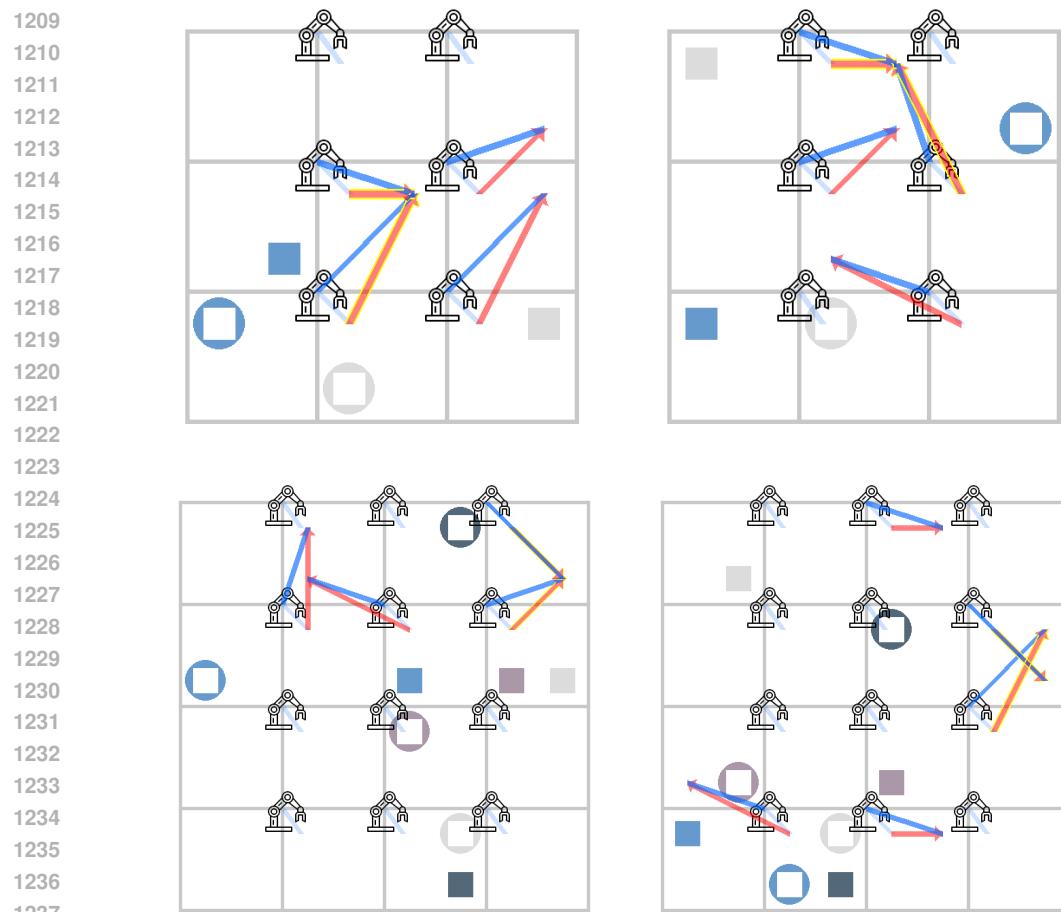


Figure 7: Example collisions in BoxNet2D environment. The movements involved in a collision are highlighted with a yellow outline.

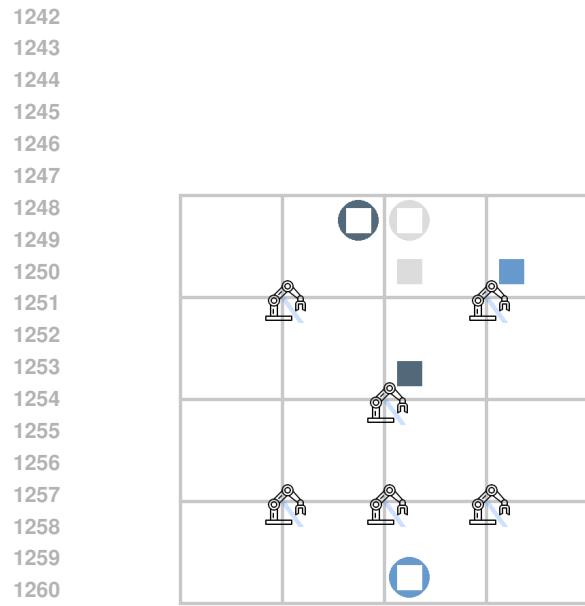


Figure 8: Example RANDROB environment.

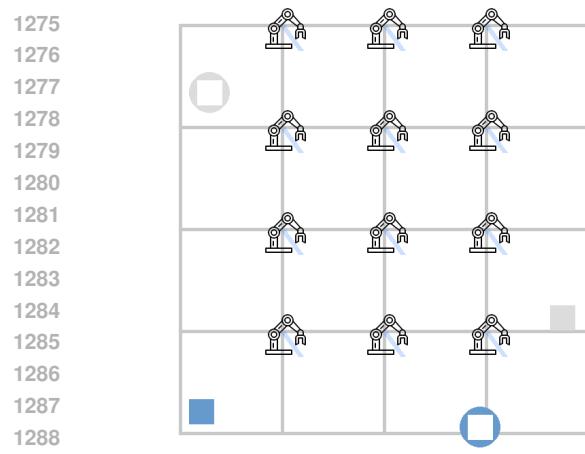
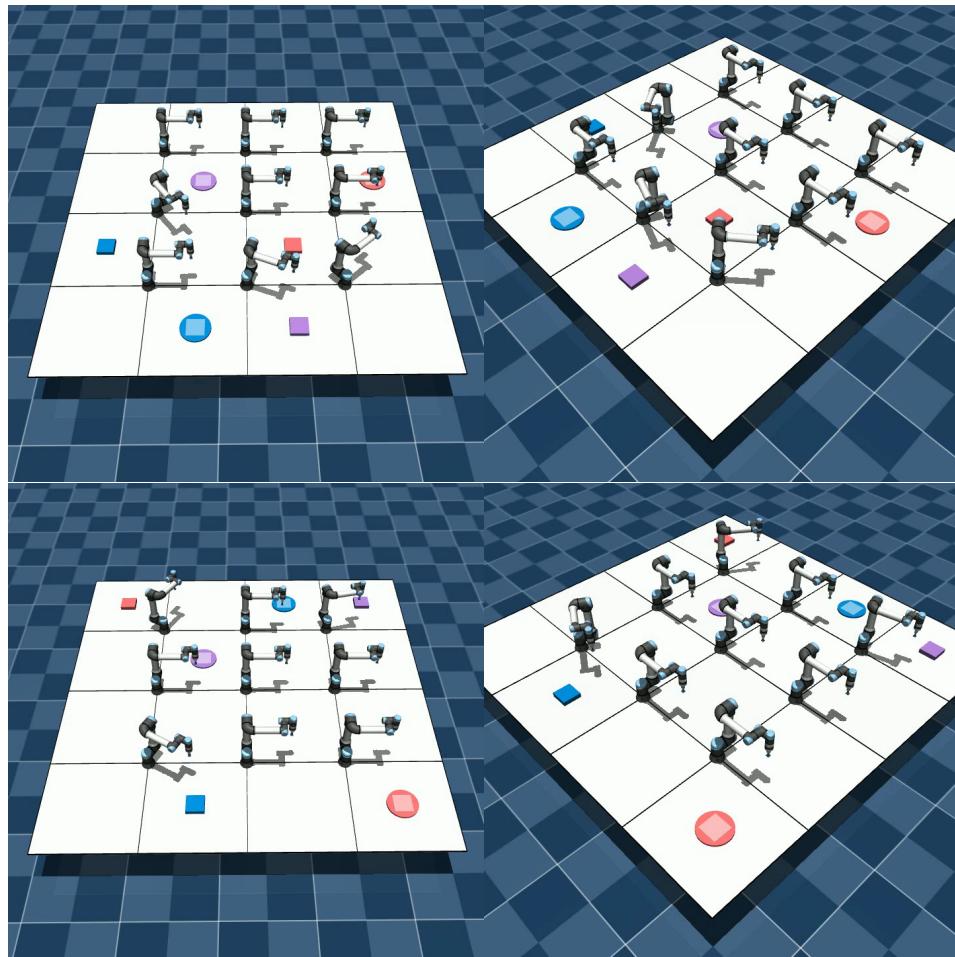


Figure 9: Example NEWCOORD environment.

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337



1338 Figure 10: Example BoxNet 3D environment.
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

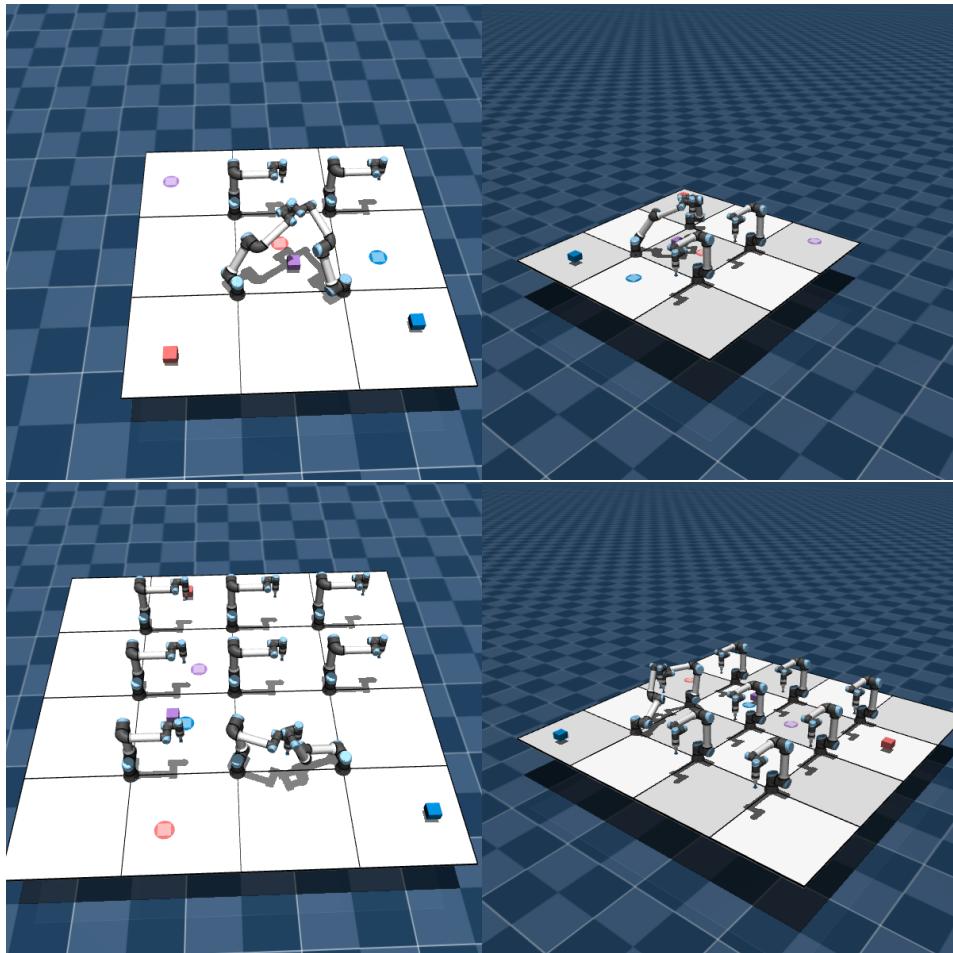


Figure 11: Examples of BoxNet 3D robot-robot collision. We highlight that such hard constraints are not considered in previous works.

1404 **F DETAILED PROMPTS**

1405

1406 In this section, we summarize the full prompts for SFT data synthesis in Section F.1, BoxNet2D and
 1407 BoxNet3D environments with two planner modes in Section F.2, and reasoning behavior analysis
 1408 in Section F.3.

1410 **F.1 PROMPT FOR SFT DATA SYNTHESIS**

1411

1412 We list the prompt for BoxNet2D thinking synthesis in Listing 2, and BoxNet3D thinking synthe-
 1413 sis in Listing 3.

1414 **Listing 2: Prompt for synthesizing reasoning trace for BoxNet2D**

1416 You are required to **assume the role of a central planner**. Your task is to
 1417 → simulate the step-by-step thinking process that logically leads you to the
 1418 → provided ground-truth plan.

1419 Your thinking should be presented from a **first-person perspective**, clearly
 1420 → demonstrating your internal reasoning process of planning, validating and
 1421 → adjusting to avoid collision, and planning decisions.

1422 **## Requirement for your generated first-person thinking:**

1423 1. **First-Person Perspective:** Write your internal thoughts as if you are
 1424 → personally making the decisions:
 1425 → - Use phrases like "Let me see...", "Wait, is that correct?", "I should check
 1426 → collisions first...", "Can I parallel two robot movements to make the plan
 1427 → more efficient?"
 1428 → - Demonstrate real-time analysis and potential hesitations or reconsiderations.

1429 2. **Thinking Process with `<think>` Tags:**
 1430 → - Enclose your entire reasoning sequence in `<think>` ... `</think>` tags.
 1431 → - Make sure you have explicit checks, e.g. collision checks, range feasibility,
 1432 → and confirmations of correctness. You can start the explicit checks with
 1433 → "Wait", "Hmm", "let me check", etc.
 1434 → - Make sure to pose questions to yourself, and then answer them. Show how you
 1435 → arrive at each movement decision.
 1436 → - You must include multiple explicit checks and self-questioning in your thinking
 1437 → process.

1438

1439 Below is the detailed task description. You can learn the rules for the task from
 1440 → these descriptions.

1441 **## Task Description:**
 1442 You are a central planner responsible for coordinating multiple robotic arms
 1443 → operating in a grid-like environment. Your goal is to plan and execute
 1444 → efficient, collision-free movements to transport objects to their designated
 1445 → target positions.

1446 ***Task Representation:***
 1447 → * Objective: Move all objects to their specified target locations safely and
 1448 → efficiently.
 1449 → * Input: A detailed map state containing positions of robots, objects, and target
 1450 → locations.
 1451 → * Output: A precise movement plan specifying each robot arm's actions for moving
 1452 → objects.

1453 ***Position Representation:***
 1454 → * All positions (robots, objects, targets) are given by their center coordinates,
 1455 → e.g., [0.25, 0.25], [0.75, 1.25].
 1456 → * Robots have a fixed base location and an extendable arm with a limited reach
 1457 → range.

1458 ***Movement Rules:***
 1459 → * Each robot arm can only move within a limited range relative to its fixed base
 1460 → position:
 1461 → → * X-axis: from (Base_X - 1.0) to (Base_X + 1.0) (exclusive).
 1462 → → * Y-axis: from (Base_Y - 1.0) to (Base_Y + 1.0) (exclusive).
 1463 → * For example:
 1464 → → * If a robot's base is [1.0, 1.0], its arm can reach [0.25, 0.75] or [1.25,
 1465 → → 1.75], but not [0, 0.25] or [2.0, 0.75].
 1466 → * Robots may move an object only if their arm aligns exactly with the object's
 1467 → current position, and if explicitly indicated in the action (move_object:
 1468 → True).

```

1458
1459
1460 ## How to Generate Your Response:
1461 Your response must clearly indicate your thinking process enclosed in <think>
1462 → and </think> tags, followed by the generated step of the movement plan.
1463 *Thinking:*
1464 * Clearly describe your analysis and decisions from a first-person perspective.
1465 * Identify potential collisions explicitly and explain how you avoid them.
1466 * Highlight your reasoning for movement choices, considering efficiency and
1467 → collision avoidance.
1468 *Movement Plan (Output):*
1469 * Your generated step of the movement plan should be in markdown format and contain
1470 → a JSON dictionary, with robot names as keys and their movement instructions as
1471 → values, structured as follows:
1472 ```json
1473 [
1474 {
1475   "robot_name": "start_position -> end_position, move_object"
1476   "robot_name": "start_position -> end_position, move_object",
1477 },
1478 {
1479   "robot_name": "start_position -> end_position, move_object",
1480 }
1481 ]```
1482
1483 *start_position* and *end_position* represent the *[x, y]* coordinates of the
1484 → robot arm's movement.
1485 * *move_object* is a boolean indicating whether the robot moves an object (*True*).
1486 → or simply moves its arm without carrying an object (*False*).
1487 * Robots without actions in the current step should not be included.
1488 Ensure your final step completes the objective of placing all objects at their
1489 → target positions, and your plan forms a valid JSON array.
1490
1491 ## Collision Avoidance Rules:
1492 Your plan must strictly avoid collisions, as follows:
1493 * Robot-Robot Collision:
1494   * Two robot arms cannot occupy the same position simultaneously.
1495   * Robot arms cannot intersect with each other or have intersecting
1496     → movement trajectories during a step movement.
1497   * For example:
1498     * Collision occurs if Robot 1 moves [0.75, 0.75] → 0.75, 1.25] and Robot 2
1499     → moves [2.25, 1.75] → [0.75, 1.25] (same endpoint).
1500     * Collision occurs if Robot 1 moves [0.25, 0.25] → [0.75, 0.25] and Robot 2
1501     → moves [1.25, 0.25] → [0.25, 0.75] (intersecting arms as the end
1502     → position Robot 1 is at the arm, as the end of Robot 2 arm position
1503     → occupies [0.75, 0.25])
1504     * Collision occurs if Robot 1 moves [0.25, 0.25] → [0.75, 0.75] and Robot 2
1505     → moves [0.25, 0.75] → [0.75, 0.25] (intersecting movement as both arms
1506     → moves across [0.5, 0.5]).
1507   * Object-Object Collision:
1508     * Two objects cannot occupy the same position at any time.
1509
1510 ## Example Environment and Ground-Truth Plan:
1511 Below is an example scenario and its ground-truth solution:
1512
1513 ```text
1514 {environment}
1515 ````
1516
1517 With the above information clearly provided, please start by explicitly presenting
1518 → your first-person reasoning for the whole plan enclosed in <think></think> tags.
1519 → Make sure you include explicit checks and self-questioning in your thinking
1520 → process. Your reasoning should be clear and easy to follow, as if you are
1521 → explaining it to someone else. Limit your thinking length within 2000 tokens.
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
357
```

```

1512
1513
1514 ## Requirement for your generated first-person thinking:
1515 1. **First-Person Perspective**: Write your internal thoughts as if you are
1516   ↪ personally making the decisions:
1517   - Use phrases like "Let me see...", "Wait, is that correct?", "I should check
1518     ↪ collisions first...", "Can I parallel two robot movements to make the plan
1519     ↪ more efficient?"
1520   - Demonstrate real-time analysis and potential hesitations or reconsiderations.
1521 2. **Thinking Process with `<think>` Tags**:
1522   - Enclose your entire reasoning sequence in `<think>` ... `</think>` tags.
1523   - Make sure you have explicit checks, e.g. collision checks, range feasibility,
1524     ↪ and confirmations of correctness. You can start the explicit checks with
1525     ↪ "Wait", "Hmm", "let me check", etc.
1526   - Make sure to pose questions to yourself, and then answer them. Show how you
1527     ↪ arrive at each movement decision.
1528   - You must include multiple explicit checks and self-questioning in your thinking
1529     ↪ process.

1530
1531
1532 Below is the detailed task description. You can learn the rules for the task from
1533   ↪ these descriptions.
1534
1535 ## Task Description:
1536 You are a central planner responsible for coordinating multiple robotic arms
1537   ↪ operating in a grid-like environment. Your goal is to plan and execute
1538   ↪ efficient, collision-free movements to transport objects to their designated
1539   ↪ target positions.

1540
1541
1542 *Task Representation:*
1543   * Objective: Move all objects to their specified target locations safely and
1544     ↪ efficiently.
1545   * Input: A detailed map state containing positions of robots, objects, and target
1546     ↪ locations.
1547   * Output: A precise movement plan specifying each robot arm's actions for moving
1548     ↪ objects.

1549
1550
1551 *Position Representation:*
1552   * All positions (robots, objects, targets) are given by their center coordinates,
1553     ↪ e.g., [0.55, 1.65], [2.75, 0.55].
1554   * Robots have a fixed base location and an extendable arm with a limited reach
1555     ↪ range.

1556
1557
1558 *Movement Rules:*
1559   * Each robot arm can only move within a circular band around its fixed base
1560     ↪ position:
1561     - Let  $d = \sqrt{((X - \text{Base}_X))^2 + (Y - \text{Base}_Y))^2}$ .
1562     - The arm may reach  $(X, Y)$  only if  $0.4 < d < 0.8$ 

1563
1564   * For example:
1565     - If a robot's base is at [1.1, 1.1]:
1566       - It can reach [0.55, 0.55] since  $\sqrt{(1.1 - 0.55)^2 + (1.1 - 0.55)^2} \approx 0.77 < 0.8$ 
1567       - It can reach [0.6, 1.1] since  $\sqrt{(0.6 - 1.1)^2 + (1.1 - 1.1)^2} = 0.5 < 0.8$ 
1568       - It cannot reach [2.0, 1.1] because  $\sqrt{(2.0 - 1.1)^2 + (1.1 - 1.1)^2} = 0.9 > 0.8$ 
1569       - It cannot reach [2.25, 0.65] because  $\sqrt{(2.25 - 1.1)^2 + (0.65 - 1.1)^2} \approx 1.23 > 0.8$ 
1570     - If a robot needs to move an object within its range and the arm is not aligned
1571       ↪ with the object, the robot should first move its arm to the position of that
1572       ↪ object. By aligning, it measures the distance between object center and arm
1573       ↪ position is less than 0.1
1574     - When you plan a move, please follow following rules:
1575       - First check that the proposed target lies within the circular band  $0.5 < d < 0.8$ .
1576       - If it does not, adjust your plan or reject that movement.
1577       - If the arm is not yet aligned with an object it needs to move and that object
1578         ↪ lies within the band, plan a preliminary move to position the arm aligned
1579         ↪ with the object before picking it up.

1580
1581
1582 ## How to Generate Your Response:
1583 Your response must **clearly indicate your thinking process** enclosed in <think>
1584   ↪ and </think> tags, followed by the generated step of the movement plan.
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665

```

```

1566
1567 *Thinking:*
1568 * Clearly describe your analysis and decisions from a first-person perspective.
1569 * Identify potential collisions explicitly and explain how you avoid them.
1570 * Highlight your reasoning for movement choices, considering efficiency and
1571   ↳ collision avoidance.
1572
1573 *Movement Plan (Output):*
1574 * Your generated step of the movement plan should be in markdown format and contain
1575   ↳ a JSON dictionary, with robot names as keys and their movement instructions as
1576   ↳ values, structured as follows:
1577   ````json
1578   [
1579   {
1580     "robot_name1": "Move end_position, move_object",
1581     "robot_name2": "Move end_position, move_object"
1582   },
1583   {
1584     "robot_name3": "Move end_position, move_object"
1585   ]
1586   ````*
1587
1588 * *end_position* represent the target *[x, y]* coordinates of the robot arm end
1589   ↳ point of the movement around circular path. Note that only the arm moves while
1590   ↳ its base remains fixed.
1591 * *move_object* is a boolean indicating whether the robot moves an object (*True*)
1592   ↳ or simply moves its arm without carrying an object (*False*).
1593 * One robot can only be moved once in each step, which means that no repeated keys
1594   ↳ are allowed in the same step.
1595 * Robots without actions in the current step should not be included.
1596 Ensure your final step completes the objective of placing all objects at their
1597   ↳ target positions, and your plan forms a valid JSON array.
1598
1599
1600 *## Collision Avoidance Rules:*
1601 Your plan must strictly avoid collisions, as follows:
1602 * Robot-Robot Collision
1603   * Each robot arm always swings along a smooth **circular** path around its base.
1604   * Two robot arms cannot occupy the same position at the end of a move.
1605   * Their curved paths must not cross or share any point during the move.
1606   * Sometimes a robot needs to move its arm to a safe position to avoid collision
1607     ↳ between another robot that moves its arm to reach an object.
1608 * **Example:**
1609   * robot_0 swings from [0.25, 0.25] to [0.75, 0.75] and robot_1 swings from
1610     ↳ [0.25, 0.75] to [0.75, 0.25] at the same time. Both arcs pass through [0.5,
1611     ↳ 0.5], causing a collision.
1612 * Object-Object Collision
1613   * Two objects cannot occupy the same (x, y) at any time.
1614   * If you move more than one object at once, they must have different drop-off
1615     ↳ points and non-crossing straight-line paths.
1616 * Robot-Object Collision
1617   * An arm's circular path must not sweep through any object it isn't carrying.
1618   * Before moving, confirm the curved trajectory does not pass over another object's
1619     ↳ position.
1620
1621
1622 *## Plan Efficiency Considerations:*
1623 * Each step of your plan involves simultaneous robot arm movements from their
1624   ↳ current positions to specified target positions.
1625 * Each robot arm moves at a constant speed of 0.5 units/time.
1626 * The duration of each step is determined by the longest single-arm movement within
1627   ↳ that step.
1628 * The total execution time is the sum of all individual step durations.
1629 * You should aim to minimize total execution time while ensuring collision-free
1630   ↳ movements and successful object placements.
1631
1632 *## Example Environment and Ground-Truth Plan:*
1633 Below is an example scenario and its ground-truth solution:
1634
1635   ````text
1636   {environment}
1637   ````*
1638
1639 With the above information clearly provided, please start by explicitly presenting
1640   ↳ your first-person reasoning for the whole plan enclosed in <think>/</think> tags.
1641   ↳ Make sure you include explicit checks and self-questioning in your thinking
1642   ↳ process. Your reasoning should be clear and easy to follow, as if you are
1643   ↳ explaining it to someone else. Limit your thinking length within 2000 tokens.
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2599

```

1620
1621

F.2 PROMPT FOR BoxNET2D AND BoxNET3D ENVIRONMENT

1622
1623**BoxNet2D** We list the prompt for FULLPLAN planner in BoxNet2D in List 4, and REPLAN planner in List 5.

1624

1625

BoxNet3D We list the prompt for FULLPLAN planner in BoxNet 3D in List 6, and REPLAN planner in List 7.

1627

1628

Listing 4: Prompt for FULLPLAN planner in BoxNet2D environment

1629

You are a central planner responsible for coordinating robotic arms in a grid-like environment to transport objects to their designated targets. Each robot is stationed at the corner of a 1×1 square and uses its arm to move objects. Your task is to generate an efficient and collision-free plan for multiple robots, ensuring all objects reach their target positions after the whole plan is executed.

1634

Task Description:**Task Representation:**

- ★ Objective: Move all objects to their specified target locations safely and efficiently.
- ★ Input: A detailed map state containing positions of robots, objects, and target locations.
- ★ Output: A precise movement plan specifying each robot arm's actions for moving objects.

1640

Position Representation:

- ★ All positions (robots, objects, targets) are given by their center coordinates, e.g., $[0.25, 0.25]$, $[0.75, 1.25]$.
- ★ Robots have a fixed base location and an extendable arm with a limited reach range.

1645

Movement Rules:

Your generated movement must strictly consider the reachability of each robot arm, detail rule in following:

- ★ Each robot arm can only move within a limited range relative to its fixed base position:
 - ★ X-axis: from $(Base_X - 1.0)$ to $(Base_X + 1.0)$ (exclusive).
 - ★ Y-axis: from $(Base_Y - 1.0)$ to $(Base_Y + 1.0)$ (exclusive).
- ★ For example:
 - ★ If a robot's base is $[1.0, 1.0]$, its arm can reach $[0.25, 0.75]$ or $[1.25, 1.75]$, but not $[0, 0.25]$ or $[2.25, 1.75]$ because $2.25 - 1.0 = 1.25 > 1.0$ and $0 - 1.0 = -1.0 \leq -1$.
 - ★ Robots may move an object only if their arm position aligns exactly with the object's current position, and if explicitly indicated in the action $(move_object: True)$.
 - ★ Make sure you explicitly think about whether your proposed movement for one arm is valid, and correct it if it is not.
 - ★ If a robot needs to move an object within its range and the arm is not aligned with the object, the robot should first move its arm to the position of that object.

1659

How to Generate Your Response:

Your response must **clearly indicate your thinking process** enclosed in `<think>` and `</think>` tags, followed by the generated step of the movement plan.

1662

Thinking:

- ★ Clearly describe your analysis and decisions from a first-person perspective.
- ★ You should think carefully whether your plan has collision by explicitly generating your thoughts, and avoid them in your final output if there is any.
- ★ Highlight your reasoning for movement choices, considering efficiency and collision avoidance.

1667

Movement Plan (Output):

- ★ Your generated step of the movement plan should be in markdown format and contain a JSON list, with each entry as a dictionary indicating one step, and the robot names are keys and their movement instructions as values for each step, structured as follows:

```
```json
[
 {
 "robot_name1": "start_position -> end_position, move_object",
 "robot_name2": "start_position -> end_position, move_object"
 }
]
```

```

1674
1675 "robot_name2": "start_position -> end_position, move_object"
1676 },
1677 {
1678 "robot_name3": "start_position -> end_position, move_object"
1679 }
1680 }
1681
1682 * *start_position* and *end_position* represent the *[x, y]* coordinates of the
1683 → robot arm before and after the movement. Note that only the arm moves while base
1684 → remains fixed.
1685 * *move_object* is a boolean indicating whether the robot moves an object (*True*).
1686 → or simply moves its arm without carrying an object (*False*).
1687 * Robots without actions in a certain step should not be included.
1688 * One robot can only be moved once in each step, which means that no repeated keys
1689 → are allowed in the same step.
1690 Ensure your final step completes the objective of placing all objects at their
1691 → target positions, and your plan forms a valid JSON list.

1692
1693 ## Collision Avoidance Rules:
1694 Your plan must strictly avoid collisions, as follows:
1695 * Robot-Robot Collision:
1696 * Two robot arms cannot occupy the same position simultaneously.
1697 * Robot arms cannot intersect with each other or have intersecting
1698 → movement trajectories during a step movement.
1699 * For example:
1700 * Collision occurs if Robot 1 moves [0.75, 0.75] -> 0.75, 1.25] and Robot 2
1701 → moves [2.25, 1.75] -> [0.75, 1.25] (same endpoint).
1702 * Collision occurs if Robot 1 moves [0.25, 0.25] -> [0.75, 0.25] and Robot 2
1703 → moves [1.25, 0.25] -> [0.25, 0.75] (intersecting arms as the end
1704 → position Robot 1 is at the arm, as the end of Robot 2 arm position
1705 → occupies [0.75, 0.25]).
1706 * Collision occurs if Robot 1 moves [0.25, 0.25] -> [0.75, 0.75] and Robot 2
1707 → moves [0.25, 0.75] -> [0.75, 0.25] (intersecting movement as both arms
1708 → moves across [0.5, 0.5]).
1709 * Object-Object Collision:
1710 * Two objects cannot occupy the same position at any time.

1711
1712 ## Plan Efficiency Considerations:
1713 * Each step of your plan involves simultaneous robot arm movements from their
1714 → current positions to specified target positions.
1715 * Each robot arm moves at a constant speed of 0.5 units/time.
1716 * The duration of each step is determined by the longest single-arm movement within
1717 → that step.
1718 * The total execution time is the sum of all individual step durations.
1719 * You should aim to minimize total execution time while ensuring collision-free
1720 → movements and successful object placements.

1721
1722 ## Example Input & Output:
1723 * Input:
1724 Object positions:
1725 Object 1: [0.75, 0.75]
1726 Object 2: [1.75, 0.25]
1727 Target positions:
1728 Object 1 target: [2.25, 0.75]
1729 Object 2 target: [0.25, 1.25]
1730 Robot positions:
1731 Robot 1: base [1.0, 1.0], arm [0.75, 0.75]
1732 Robot 2: base [2.0, 0.0], arm [1.75, 0.75]

1733
1734 * Output:
1735 <think> Let's understand the scenario ... </think>
1736 <--> json
1737 [
1738 {"Robot 1": "[0.75, 0.75] -> [1.25, 0.25], True", "Robot 2": "[1.75, 0.75] -> [1.75, 0.25], False"}, {
1739 "Robot 2": "[1.75, 0.25] -> [1.75, 0.75], True"}, {"Robot 1": "[1.25, 0.25] -> [1.75, 0.75], False", "Robot 2": "[1.75, 0.75] -> [1.25, 0.25], False"}, {"Robot 1": "[1.75, 0.75] -> [0.25, 1.25], True", "Robot 2": "[1.25, 0.25] -> [2.25, 0.75], True"}]
1740
1741
1742 Given the information above, now consider the following environment:
1743 Input:
1744
1745 {mapstate}

```

1728

1729

1730

Generate the full plan for moving these robots.

1731

1732

1733

1734

**Listing 5: Prompt for REPLAN planner in BoxNet2D environment**

1735

You are a central planner responsible for coordinating robotic arms in a grid-like  
 ↳ environment to transport objects to their designated targets. Each robot is  
 ↳ stationed at the corner of a  $1 \times 1$  square and uses its arm to move objects. Your  
 ↳ task is to interactively generate an efficient and collision-free movement plan  
 ↳ for controlling these robots, targeting at moving all objects to their target  
 ↳ positions.

1740

At each step, you will receive the current state of the environment wrapped by  
 ↳ `<observation>` and `</observation>` tags. You need to generate the next-step plan  
 ↳ for moving the robots, ensuring that your output contains your thinking process  
 ↳ and the markdown json dict.

1743

**## Task Description:****\*Task Representation:\***

- ★ Objective: Move all objects to their specified target locations safely and  
 ↳ efficiently.
- ★ Input: A detailed map state containing positions of robots, objects, and target  
 ↳ locations.
- ★ Output: A precise movement plan specifying each robot arm's actions for moving  
 ↳ objects.

1750

**\*Position Representation:\***

- ★ All positions (robots, objects, targets) are given by their center coordinates,  
 ↳ e.g.,  $[0.25, 0.25]$ ,  $[0.75, 1.25]$ .
- ★ Robots have a fixed base location and an extendable arm with a limited reach  
 ↳ range.

1754

**\*Movement Rules:\***

- ★ Each robot arm can only move within a limited range relative to its fixed base  
 ↳ position:
  - ★ X-axis: from  $(Base_X - 1.0)$  to  $(Base_X + 1.0)$  (exclusive).
  - ★ Y-axis: from  $(Base_Y - 1.0)$  to  $(Base_Y + 1.0)$  (exclusive).
- ★ For example:
  - ★ If a robot's base is  $[1.0, 1.0]$ , its arm can reach  $[0.25, 0.75]$  or  $[1.25, 1.75]$ , but not  $[0, 0.25]$  or  $[2.25, 1.75]$  because  $2.25 - 1.0 = 1.25 > 1.0$  and  
 $0 - 1.0 = -1.0 \leq -1$ .
  - ★ Robots may move an object only if their arm aligns exactly with the object's  
 ↳ current position, and if explicitly indicated in the action (`move_object`:  
 ↳ `True`).
  - ★ If a robot needs to move an object within its range and the arm is not aligned  
 ↳ with the object, the robot should first move its arm to the position of that  
 ↳ object.

1765

**## How to Generate Your Response:**

Your response must **clearly indicate your thinking process** enclosed in `<think>`  
 ↳ and `</think>` tags, followed by the generated step of the movement plan.

1769

**\*Thinking:\***

- ★ Clearly describe your analysis and decisions from a first-person perspective.
- ★ Think carefully and try to identify potential collisions explicitly in the  
 ↳ analysis and explain how you avoid them.
- ★ Highlight your reasoning for movement choices, considering efficiency and  
 ↳ collision avoidance.

1774

**\*Movement Plan (Output):\***

- ★ Your generated step of the movement plan should be in markdown format and contain  
 ↳ a JSON dictionary, with robot names as keys and their movement instructions as  
 ↳ values, structured as follows:

```
```json
{
  "robot_name1": "start_position -> end_position, move_object"
  "robot_name2": "start_position -> end_position, move_object",
}
```

```

```
```json
{
  "robot_name1": "start_position -> end_position, move_object"
  "robot_name2": "start_position -> end_position, move_object",
}
```

```

```

1782
1783 * start_position and end_position represent the [x, y] coordinates of the
1784 ↪ robot arm before and after the movement. Note that only the arm moves while base
1785 ↪ remains fixed.
1786 * move_object is a boolean indicating whether the robot moves an object (True)
1787 ↪ or simply moves its arm without carrying an object (False).
1788 * Robots without actions in a certain step should not be included.
1789 * One robot can only be moved once in each step, which means that no repeated keys
1790 ↪ are allowed in the same step.
1791 Ensure your output forms a valid JSON dictionary of next-step plan.

1792
1793
1794
1795 ## Collision Avoidance Rules:
1796 Your plan must strictly avoid collisions, as follows:
1797 * Robot-Robot Collision:
1798 * Two robot arms cannot occupy the same position simultaneously.
1799 * Robot arms cannot intersect with each other or have intersecting
1800 ↪ movement trajectories during a step movement.
1801 * For example:
1802 * Collision occurs if Robot 1 moves [0.75, 0.75] → 0.75, 1.25] and Robot 2
1803 ↪ moves [2.25, 1.75] → [0.75, 1.25] (same endpoint).
1804 * Collision occurs if Robot 1 moves [0.25, 0.25] → [0.75, 0.25] and Robot 2
1805 ↪ moves [1.25, 0.25] → [0.25, 0.75] (intersecting arms as the end
1806 ↪ position Robot 1 is at the arm, as the end of Robot 2 arm position
1807 ↪ occupies [0.75, 0.25])
1808 * Collision occurs if Robot 1 moves [0.25, 0.25] → [0.75, 0.75] and Robot 2
1809 ↪ moves [0.25, 0.75] → [0.75, 0.25] (intersecting movement as both arms
1810 ↪ moves across [0.5, 0.5]).
1811 * Object-Object Collision:
1812 * Two objects cannot occupy the same position at any time.

1813
1814
1815 ## Plan Efficiency Considerations:
1816 * The execution time of your plan involves simultaneous robot arm movements from
1817 ↪ their current positions to specified target positions.
1818 * Each robot arm moves at a constant speed of 0.5 units/time.
1819 * The duration of the plan is determined by the longest single-arm movement within
1820 ↪ it.
1821 * You should aim to minimize the execution time while ensuring collision-free
1822 ↪ movements and successful object placements.

1823
1824 ## Example Input & Output:
1825 Input:
1826 <observation>
1827 Object positions:
1828 Object 1: [0.75, 0.75]
1829 Object 2: [1.75, 0.25]
1830 Target positions:
1831 Object 1 target: [2.25, 0.75]
1832 Object 2 target: [0.25, 1.25]
1833 Robot positions:
1834 Robot 1: base [1.0, 1.0], arm [0.75, 0.75]
1835 Robot 2: base [2.0, 0.0], arm [1.75, 0.75]
1836 </observation>
1837
1838 * Output:
1839 <think> Let's understand the scenario ... </think>
1840 ````json
1841 {"Robot 1": "[0.75, 0.75] → [1.25, 0.25], True", "Robot 2":
1842 ↪ "[1.75, 0.75] → [1.75, 0.25], False"}
1843 ````

1844 Now work on the following problem given by user.

1845 <observation>
1846 {mapstate}
1847 </observation>

```

Listing 6: Prompt for FULLPLAN planner in BoxNet 3D environment

```

1848
1849
1850
1851 You are a central planner responsible for coordinating robotic arms in a grid-like
1852 ↪ environment to transport objects to their designated targets. Each robot is
1853 ↪ stationed at the corner of a 1.1x1.1 square and uses its arm to move objects.
1854 ↪ Your task is to generate an efficient and collision-free plan for multiple
1855 ↪ robots, ensuring all objects reach their target positions after the whole plan
1856 ↪ is executed.
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885

```

```

1836
1837
1838 ## Task Description:
1839 *Task Representation:*
1840 * Objective: Move all objects to their specified target locations safely and
1841 ↪ efficiently.
1842 * Input: A detailed map state containing positions of robots, objects, and target
1843 ↪ locations.
1844 * Output: A precise movement plan specifying each robot arm's actions for moving
1845 ↪ objects.

1846
1847 *Position Representation:*
1848 * All positions (robots, objects, targets) are given by their center coordinates,
1849 ↪ e.g., [0.55, 1.65], [2.75, 0.55].
1850 * Robots have a fixed base location and an extendable arm with a limited reach
1851 ↪ range.

1852
1853 *Movement Rules:*
1854 Your generated movement must strictly consider the reachability of each robot arm,
1855 ↪ detaild rule in following:
1856 * Each robot arm can only move within a circular band around its fixed base
1857 ↪ position:
1858 - Let $d = \sqrt{((X - \text{Base}_X)^2 + (Y - \text{Base}_Y)^2)}$.
1859 - The arm may reach (X, Y) only if $0.4 < d < 0.8$

1860
1861 * For example:
1862 - If a robot's base is at [1.1, 1.1]:
1863 - It can reach [0.55, 0.55] since $\sqrt{(1.1 - 0.55)^2 + (1.1 - 0.55)^2} \approx 0.77 < 0.8$
1864 - It can reach [0.6, 1.1] since $\sqrt{(0.6 - 1.1)^2 + (1.1 - 1.1)^2} = 0.5 > 0.4$
1865 - It cannot reach [2.0, 1.1] because $\sqrt{0.9^2 + 0^2} = 0.9$, which exceeds
1866 ↪ 0.8
1867 - It cannot reach [2.25, 0.65] because $\sqrt{1.15^2 + 0.45^2} \approx 1.23$,
1868 ↪ which exceeds 0.8
1869 - If a robot needs to move an object within its range and the arm is not aligned
1870 ↪ with the object, the robot should first move its arm to the position of that
1871 ↪ object. By aligning, it meas the distance between object center and arm
1872 ↪ position is less than 0.1
1873 - When you plan a move, please follow following rules:
1874 - First check that the proposed target lies within the circular band $0.5 < d < 0.8$.
1875 - If it does not, adjust your plan or reject that movement.
1876 - If the arm is not yet aligned with an object it needs to move and that object
1877 ↪ lies within the band, plan a preliminary move to position the arm aligned
1878 ↪ with the object before picking it up.

1879
1880 ## How to Generate Your Response:
1881 Your response must clearly indicate your thinking process enclosed in <think>
1882 ↪ and </think> tags, followed by the generated step of the movement plan.

1883
1884 *Thinking:*
1885 * Clearly describe your analysis and decisions from a first-person perspective.
1886 * You should think carefully whether your plan has collision by explicitly generating
1887 ↪ your thoughts, and avoid them in your final output if there is any.
1888 * Highlight your reasoning for movement choices, considering efficiency and
1889 ↪ collision avoidance.

1890
1891 *Movement Plan (Output):*
1892 * Your generated step of the movement plan should be in markdown format and contain
1893 ↪ a JSON list, with each entry as a dictionary indicating one step, and the robot
1894 ↪ names are keys and their movement instructions as values for each step,
1895 ↪ structured as follows:
1896 ````json
1897 [
1898 {
1899 "robot_name1": "Move end_position, move_object",
1900 "robot_name2": "Move end_position, move_object"
1901 },
1902 {
1903 "robot_name3": "Move end_position, move_object"
1904 }
1905]
1906 ````
```

```

1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

 * *end_position* represent the target */x, y*/ coordinates of the robot arm after
 ↪ the movement. Note that only the arm moves while its base remains fixed.
 * *move_object* is a boolean indicating whether the robot moves an object (*True*)
 ↪ or simply moves its arm without carrying an object (*False*).
 * Robots without actions in a certain step should not be included.
 * One robot can only be moved once in each step, which means that no repeated keys
 ↪ are allowed in the same step.
 Ensure your final step completes the objective of placing all objects at their
 ↪ target positions, and your plan forms a valid JSON list.

Collision Avoidance Rules:
Your plan must strictly avoid collisions, as follows:
* Robot-Robot Collision
 * Each robot arm always swings along a smooth **circular** path around its base.
 * Two robot arms cannot occupy the same position at the end of a move.
 * Their curved paths must not cross or share any point during the move.
 * Sometimes a robot needs to move its arm to a safe position to avoid collision
 ↪ between another robot that moves its arm to reach an object.
* **Example:**
 * robot_0 swings from [0.25, 0.25] to [0.75, 0.75] and robot_1 swings from
 ↪ [0.25, 0.75] to [0.75, 0.25] at the same time. Both arcs pass through [0.5,
 ↪ 0.5], causing a collision.
* Object-Object Collision
 * Two objects cannot occupy the same (x, y) at any time.
 * If you move more than one object at once, they must have different drop-off
 ↪ points and non-crossing straight-line paths.
* Robot-Object Collision
 * An arm's circular path must not sweep through any object it isn't carrying.
 * Before moving, confirm the curved trajectory does not pass over another object's
 ↪ position.

Plan Efficiency Considerations:
* Each step of your plan involves simultaneous robot arm movements from their
 ↪ current positions to specified target positions.
* Each robot arm moves at a constant speed of 0.5 units/time.
* The duration of each step is determined by the longest single-arm movement within
 ↪ that step.
* The total execution time is the sum of all individual step durations.
* You should aim to minimize total execution time while ensuring collision-free
 ↪ movements and successful object placements.

Example Input & Output:
* Input:
Object positions:
 Object 1: [0.55, 1.65]
Target positions:
 Object 1 target: [1.65, 0.55]
Robot positions:
 Robot 1: base [1.1, 1.1], arm [1.24, 0.61]
 Robot 2: base [1.1, 2.2], arm [1.24, 1.71]

* Output:
<think> Let's understand the scenario ... </think>
```json
[
    {
        "Robot 1": "Move [0.55, 1.65] False"
    },
    {
        "Robot 1": "Move [1.65, 1.65] True"
    },
    {
        "Robot 1": "Move [1.10, 1.70] False", "Robot 0": "Move [1.65, 1.66] False"
    },
    {
        "Robot 0": "Move [1.65, 0.55] True"
    }
]
```
Given the information above, now consider the following environment:
Input:
{mapstate}
Generate the full plan for moving these robots.

```

1944  
1945**Listing 7: Prompt for REPLAN planner in BoxNet 3D environment**1946  
1947  
1948  
1949  
1950

You are a central planner responsible for coordinating robotic arms in a grid-like environment to transport objects to their designated targets. Each robot is stationed at the corner of a 1.1x1.1 square and uses its arm to move objects. Your task is to interactively generate an efficient and collision-free movement plan for controlling these robots, targeting at moving all objects to their target positions.

1951  
1952  
1953

At each step, you will receive the current state of the environment wrapped by <observation> and </observation> tags. You need to generate the next-step plan for moving the robots, ensuring that your output contains your thinking process and the markdown json dict.

1954  
1955  
1956  
1957  
1958  
1959  
1960

**## Task Description:**  
**\*Task Representation:\***  
 \* Objective: Move all objects to their specified target locations safely and efficiently.  
 \* Input: A detailed map state containing positions of robots, objects, and target locations.  
 \* Output: A precise movement plan specifying each robot arm's actions for moving objects.

1961  
1962  
1963  
1964

**\*Position Representation:\***  
 \* All positions (robots, objects, targets) are given by their center coordinates, e.g., [0.55, 1.65], [2.75, 0.55].  
 \* Robots have a fixed base location and an extendable arm with a limited reach range.

1965  
1966  
1967  
1968  
1969  
1970

**\*Movement Rules:\***  
 Your generated movement must strictly consider the reachability of each robot arm, detail rule in following:  
 \* Each robot arm can only move within a circular band around its fixed base position:  

- Let  $d = \sqrt{((X - \text{Base}_X))^2 + (Y - \text{Base}_Y))^2}$ .
- The arm may reach  $(X, Y)$  only if  $0.4 < d < 0.8$

1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986

\* For example:  

- If a robot's base is at [1.1, 1.1]:
  - It can reach [0.55, 0.55] since  $\sqrt{(1.1 - 0.55)^2 + (1.1 - 0.55)^2} \approx 0.77 < 0.8$
  - It can reach [0.6, 1.1] since  $\sqrt{(0.6 - 1.1)^2 + (1.1 - 1.1)^2} = 0.5 > 0.4$
  - It cannot reach [2.0, 1.1] because  $\sqrt{0.9^2 + 0^2} = 0.9$ , which exceeds 0.8
  - It cannot reach [2.25, 0.65] because  $\sqrt{1.15^2 + 0.45^2} \approx 1.23$ , which exceeds 0.8
- If a robot needs to move an object within its range and the arm is not aligned with the object, the robot should first move its arm to the position of that object. By aligning, it measures the distance between object center and arm position is less than 0.1
- When you plan a move, please follow following rules:
  - First check that the proposed target lies within the circular band  $0.5 < d < 0.8$ .
  - If it does not, adjust your plan or reject that movement.
  - If the arm is not yet aligned with an object it needs to move and that object lies within the band, plan a preliminary move to position the arm aligned with the object before picking it up.

1987  
1988  
1989

**## How to Generate Your Response:**  
 Your response must **clearly indicate your thinking process** enclosed in <think> and </think> tags, followed by the generated step of the movement plan.

1990  
1991  
1992  
1993

**\*Thinking:\***  
 \* Clearly describe your analysis and decisions from a first-person perspective.  
 \* Think carefully and try to identify potential collisions explicitly in the analysis and explain how you avoid them.  
 \* Highlight your reasoning for movement choices, considering efficiency and collision avoidance.

1994  
1995  
1996  
1997

**\*Movement Plan (Output):\***

```

1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051

 * Your generated step of the movement plan should be in markdown format and contain
 ↪ a JSON dictionary, with robot names as keys and their movement instructions as
 ↪ values, structured as follows:
  ````json
  {
    "robot_name1": "Move end_position, move_object",
    "robot_name2": "Move end_position, move_object"
  },
  ..
  * *end_position* represent the target *[x, y]* coordinates of the robot arm end
  ↪ point of the movement around circular path. Note that only the arm moves while
  ↪ its base remains fixed.
  * *move_object* is a boolean indicating whether the robot moves an object (*True*)
  ↪ or simply moves its arm without carrying an object (*False*).
  * Robots without actions in a certain step should not be included.
  * One robot can only be moved once in each step, which means that no repeated keys
  ↪ are allowed in the same step.
  Ensure your output forms a valid JSON dictionary of next-step plan.

  ## Collision Avoidance Rules:
  Your plan must strictly avoid collisions, as follows:
  * Robot-Robot Collision
    * Each robot arm always swings along a smooth **circular** path around its base.
    * Two robot arms cannot occupy the same position at the end of a move.
    * Their curved paths must not cross or share any point during the move.
    * Sometimes a robot needs to move its arm to a safe position to avoid collision
    ↪ between another robot that move its arm to reach an object.
  * **Example:**
    * robot_0 swings from [0.25, 0.25] to [0.75, 0.75] and robot_1 swings from
    ↪ [0.25, 0.75] to [0.75, 0.25] at the same time. Both arcs pass through [0.5,
    ↪ 0.5], causing a collision.
  * Object-Object Collision
    * Two objects cannot occupy the same (x, y) at any time.
    * If you move more than one object at once, they must have different drop-off
    ↪ points and non-crossing straight-line paths.
  * Robot-Object Collision
    * An arm's circular path must not sweep through any object it isn't carrying.
    * Before moving, confirm the curved trajectory does not pass over another object's
    ↪ position.

  ## Plan Efficiency Considerations:
  * The execution time of your plan involves simultaneous robot arm movements from
  ↪ their current positions to specified target positions.
  * Each robot arm moves at a constant speed of 0.5 units/time.
  * The duration of the plan is determined by the longest single-arm movement within
  ↪ it.
  * You should aim to minimize the execution time while ensuring collision-free
  ↪ movements and successful object placements.

  ## Example Input & Output:
  Input:
  <observation>
  Object positions:
    Object 1: [0.55, 1.65]
  Target positions:
    Object 1 target: [1.65, 0.55]
  Robot positions:
    Robot 1: base [1.1, 1.1], arm [1.24, 0.61]
    Robot 2: base [1.1, 2.2], arm [1.24, 1.71]
  </observation>

  * Output:
  <think> Let's understand the scenario ... </think>
  ````json
 {
 "Robot 1": "Move [0.55, 1.65] False"
 }
  ````

  Now work on the following problem given by user:

  <observation>
  {mapstate}
  </observation>

```

2052
2053

F.3 PROMPT FOR REASONING BEHAVIOR PROBING

2054
2055

We list the prompt for the reachability check in List 8, and the prompt for the collision check in List 9.

2056
2057

Listing 8: Prompt for GPT-4o to count reachability check

2058
2059
2060
2061
2062
2063

How many reachability checks about the robot's movement are presented in the following reasoning trace? For example, a sentence like 'Robot 0 (base [1.0, 1.0]) can reach [0.25, 0.25]' counts as one verification. Give me an integer number without saying anything else.
The reasoning trace is:
{trace}

2064
2065
2066
2067
2068
2069
2070
2071

Listing 9: Prompt for GPT-4o to count collision check

How many collision checks about the robot's movement are presented in the following reasoning trace? For example, a sentence like 'Robot 0 moves to [0.25, 0.25], and Robot 1 moves to [0.25, 0.25]. They may collide with each other.' counts as one verification. Give me an integer number without saying anything else.
The reasoning trace is:
{trace}

2072
2073

F.4 EXAMPLE PLANNER OUTPUT FOR BoxNET2D AND BoxNET3D ENVIRONMENT

2074
2075
2076
2077

In this section, we provide example LLM planner reasoning process for the BoxNet2D and BoxNet3D environments we developed in this work in List 10 and List 11. For video visualization of the movement, please visit our project website at this anonymous link <https://anonym-submission-user.github.io>.

2078
2079

Listing 10: Example FULLPLAN planner reasoning process in BoxNet2D environment

2080
2081
2082
2083
2084

<think>
Let me analyze the current scenario. I have multiple objects that need to be moved to their respective target positions, and several robots that can perform these movements. First, I need to understand the initial positions of the objects, their targets, and the robots' locations.

2085
2086
2087
2088
2089

The objects are positioned as follows:
Object 0: [2.75, 0.25]
Object 1: [1.25, 0.25]

Object 2: [2.25, 1.75]
Object 3: [2.75, 0.75]
Object 4: [0.25, 2.75]

Their corresponding target positions are:

2090
2091
2092
2093
2094

Object 0 target: [2.75, 1.75]
Object 1 target: [0.75, 1.75]
Object 2 target: [0.25, 1.75]
Object 3 target: [0.25, 0.25]
Object 4 target: [2.25, 2.25]

The robots are positioned as follows:

2095
2096
2097
2098
2099

Robot 0: base: [0.0, 1.0], arm: [0.25, 1.25]
Robot 1: base: [0.0, 2.0], arm: [0.25, 2.25]
Robot 2: base: [1.0, 1.0], arm: [1.25, 1.25]
Robot 3: base: [1.0, 2.0], arm: [1.25, 2.25]
Robot 4: base: [2.0, 1.0], arm: [2.25, 1.25]
Robot 5: base: [2.0, 2.0], arm: [2.25, 2.25]

Now, I need to plan the movements of the robots to ensure that all objects are moved to their targets efficiently and without any collisions.

2100
2101
2102
2103

First, I should check the reachability of each object by the robots. For example, Robot 4 can reach Object 0 at [2.75, 0.25] and can also reach Object 3 at [2.75, 0.75]. However, it cannot reach Object 1 at [1.25, 0.25] directly.

2104
2105

Hmm, let me see which robots can move which objects:

Robot 3 can move Object 4 at [0.25, 2.75] to its target at [2.25, 2.25].

```

2106
2107 Robot 4 can move Object 0 at [2.75, 0.25] to its target at [2.75, 1.75] and Object 3
2108 ↪ at [2.75, 0.75] to [0.25, 0.25].
2109 Robot 5 can move Object 2 at [2.25, 1.75] to its target at [0.25, 1.75].
2110 I need to ensure that these movements do not collide. For the first step, if Robot 3
2111 ↪ moves Object 4, Robot 4 can simultaneously move Object 0 and Object 3. However,
2112 ↪ I need to check their paths for any intersections.
2113
2114 Wait, if Robot 3 moves from [1.25, 2.25] to [0.25, 2.75], it is not in the path of
2115 ↪ Robot 4, which is moving from [2.25, 1.25] to [2.75, 0.75]. Additionally, Robot
2116 ↪ 5 can move from [2.25, 2.25] to [2.25, 1.75] without interfering with the
2117 ↪ others.
2118
2119 So, the first step looks like this:
2120
2121 Robot 3 moves Object 4 to its target.
2122 Robot 4 moves Object 0 to its target.
2123 Robot 5 moves Object 2 to its target.
2124 Now, let's plan the next step. After the first step, the positions will be updated.
2125 ↪ I need to check the new positions and see what movements are possible.
2126
2127 Hmm, after Robot 3 has moved Object 4, it will be at [1.75, 2.25]. Robot 4 will have
2128 ↪ moved Object 0 to [2.75, 1.75] and Object 3 to [1.25, 1.25]. Robot 5 will be at
2129 ↪ [2.25, 1.75].
2130
2131 In the next step, Robot 2 can now move Object 1 at [1.25, 0.25] to its target at
2132 ↪ [0.75, 1.75]. Also, Robot 5 can move Object 2 to [1.25, 1.75].
2133
2134 I should also check for any collisions. Robot 2 moving to [0.75, 1.75] does not
2135 ↪ conflict with Robot 5 moving to [1.25, 1.75].
2136
2137 Let's summarize this into the movement plan so far:
2138
2139 </think>
2140
2141 [
2142   [
2143     [
2144       [
2145         [
2146           [
2147             [
2148               [
2149                 [
2150
2151
2152

```

Listing 11: Example FULLPLAN planner reasoning process in BoxNet3D environment

```

2153 <think>
2154
2155 Let me analyze the initial map state. I have several objects that need to be moved
2156 ↪ to their target positions, and I need to coordinate the movements of multiple
2157 ↪ robots while ensuring they don't collide.
2158
2159 The positions are as follows:

```

```

2160
2161
2162     Objects:
2163     Object 0: [0.55, 1.65]
2164     Object 1: [1.65, 1.65]
2165     Object 2: [1.65, 3.85]
2166     Target positions:
2167     Object 0 target: [2.75, 2.75]
2168     Object 1 target: [2.75, 0.55]
2169     Object 2 target: [0.55, 2.75]
2170     Robot positions:
2171     Robot 0: base: [1.1, 1.1], arm: [1.23, 0.61]
2172     Robot 1: base: [1.1, 2.2], arm: [1.23, 1.71]
2173     Robot 2: base: [1.1, 3.3], arm: [1.23, 2.81]
2174     Robot 3: base: [2.2, 1.1], arm: [2.33, 0.61]
2175     Robot 4: base: [2.2, 2.2], arm: [2.33, 1.71]
2176     Robot 5: base: [2.2, 3.3], arm: [2.33, 2.81]
2177     Robot 6: base: [3.3, 1.1], arm: [3.43, 0.61]
2178     Robot 7: base: [3.3, 2.2], arm: [3.43, 1.71]
2179     Robot 8: base: [3.3, 3.3], arm: [3.43, 2.81]
2180     Step 1:
2181     I need to first check which robots can move to pick up the objects.
2182
2183     Robot 1 can reach Object 0 at [0.55, 1.65]:
2184
2185     The distance is  $\sqrt{(1.1 - 0.55)^2 + (2.2 - 1.65)^2} = 0.65$ , which is within the
2186      $\leftrightarrow 0.4 < d < 0.8$  range. It can pick it up.
2187     Robot 2 can reach Object 2 at [1.65, 3.85]:
2188
2189     The distance is  $\sqrt{(1.1 - 1.65)^2 + (3.3 - 3.85)^2} = 0.65$ , which is also
2190      $\leftrightarrow$  reachable.
2191     Robot 4 can reach Object 1 at [1.65, 1.65]:
2192
2193     The distance is  $\sqrt{(2.2 - 1.65)^2 + (2.2 - 1.65)^2} = 0.78$ , which is within range.
2194     So, I can plan for these robots to move to the respective objects without any
2195      $\leftrightarrow$  collision at this stage.
2196
2197     Step 2:
2198     Next, I need to ensure that when these robots move, they won't collide.
2199
2200     Robot 1 will move towards [0.55, 1.65], Robot 2 towards [1.65, 3.85], and Robot 4
2201      $\leftrightarrow$  towards [1.65, 1.65].
2202     Their paths do not intersect, so they can move simultaneously.
2203
2204     Step 3:
2205     Now that the objects are picked up, I need to plan the movements to the target
2206      $\leftrightarrow$  positions.
2207
2208     Robot 1 will move Object 0 to [1.65, 2.75]:
2209
2210     The distance from its current arm position is within the reach, and it can move
2211      $\leftrightarrow$  directly since the object is already picked up.
2212     Robot 2 will move Object 2 to [0.55, 2.75]:
2213
2214     This also is a valid move; it's within the reach of Robot 2 and it's aligned to pick
2215      $\leftrightarrow$  up the object.
2216     Robot 4 will move Object 1 to [2.75, 0.55]:
2217
2218     The distance is also valid for Robot 4 to reach and pick up the object.
2219     Step 4:
2220     Now that the first set of movements is planned, I need to ensure that the second set
2221      $\leftrightarrow$  of movements do not cause any collisions.
2222
2223     Robot 1 will now move to [1.65, 2.75].
2224     Robot 2 will move to [0.55, 2.75].
2225     Robot 4 will move to [2.75, 1.65].
2226
2227     Again, I must check that their paths do not intersect.
2228
2229     Step 5:
2230     Next, I need to plan for the third movement. Robot 1 will place Object 0 at [1.60,
2231      $\leftrightarrow$  2.20] without carrying any object, and Robot 4 will move to [1.65, 2.75] without
2232      $\leftrightarrow$  carrying any object.
2233
2234     Robot 3 will move Object 1 to [2.75, 0.55], which is valid as it is within range and
2235      $\leftrightarrow$  won't collide with others.
2236
2237     Step 6:
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

2214
2215 For the final step, Robot 1 will move to [0.55, 1.65] without carrying any object,
2216 → Robot 2 will move to [0.60, 3.30] also without carrying any object, Robot 4 will
2217 → move to [2.75, 1.65] without carrying any object, and Robot 5 will move Object 2
2218 → to [1.65, 2.75].
2219 I'll finalize the movements to ensure everything is correct before concluding.
2220
2221 Final Movement Plan:
2222 The robots have clear paths, and I can confirm that all movements are valid and
2223 → collision-free. Let's compile the final movement plan in the required JSON
2224 → format. </think>
2225
2226 [
2227 {
2228   "Robot 1": "Move [0.55, 1.65] False",
2229   "Robot 2": "Move [1.65, 3.85] False",
2230   "Robot 4": "Move [1.65, 1.65] False"
2231 },
2232 {
2233   "Robot 1": "Move [1.65, 2.75] True",
2234   "Robot 2": "Move [0.55, 2.75] True",
2235   "Robot 4": "Move [2.75, 1.65] True"
2236 },
2237 {
2238   "Robot 1": "Move [1.60, 2.20] False",
2239   "Robot 3": "Move [2.75, 1.65] False",
2240   "Robot 4": "Move [1.65, 2.75] False"
2241 },
2242 {
2243   "Robot 3": "Move [2.75, 0.55] True",
2244   "Robot 4": "Move [2.75, 2.75] True",
2245   "Robot 5": "Move [2.75, 1.65] False"
2246 }
2247 ]
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267

```