

RoboHorizon: An LLM-Assisted Multi-View World Model for Long-Horizon Robotic Manipulation

Zixuan Chen¹, Jing Huo¹, Yangtao Chen¹ and Yang Gao¹

¹State Key Laboratory for Novel Software Technology, Nanjing University, China

{chenzx, huojing}@nju.edu.cn, 502023330009@smail.nju.edu.cn, gaoy@nju.edu.cn

Abstract

Efficient control in long-horizon robotic manipulation is challenging due to complex representation and policy learning requirements. Model-based visual reinforcement learning (RL) has shown great potential in addressing these challenges but still faces notable limitations, particularly in handling sparse rewards and complex visual features in long-horizon environments. To address these limitations, we propose the *Recognize-Sense-Plan-Act (RSPA)* pipeline for long-horizon tasks and further introduce

RoboHorizon, an LLM-assisted multi-view world model tailored for long-horizon robotic manipulation. In RoboHorizon, pre-trained LLMs generate dense reward structures for multi-stage sub-tasks based on task language instructions, enabling robots to better *recognize* long-horizon tasks. Keyframe discovery is then integrated into the multi-view masked autoencoder (MAE) architecture to enhance the robot’s ability to *sense* critical task sequences, strengthening its multi-stage perception of long-horizon processes. Leveraging these dense rewards and multi-view representations, a robotic world model is constructed to efficiently *plan* long-horizon tasks, enabling the robot to reliably *act* through RL algorithms. Experiments on two representative benchmarks, RLBench and FurnitureBench, show that RoboHorizon outperforms state-of-the-art visual model-based RL methods, achieving a 23.35% improvement in task success rates on RLBench’s 4 short-horizon tasks and a 29.23% improvement on 6 long-horizon tasks from RLBench and 3 furniture assembly tasks from FurnitureBench.

1 Introduction

A general-purpose robotic manipulator for real-life applications should be capable of performing long-horizon tasks composed of multiple sub-task phases, such as kitchen organization or warehouse picking. For instance, kitchen organization requires a robot to complete tasks like sorting food items, placing them into the refrigerator, and cleaning the countertops, while warehouse picking might involve identifying orders, picking items, and packing them. But how

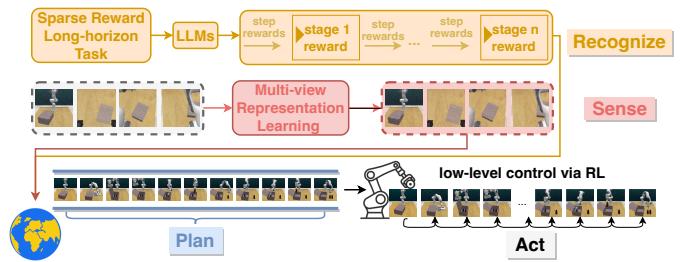


Figure 1: The proposed **RSPA pipeline** for long-horizon robotic manipulation.

can we design such a comprehensive robotic system? Traditionally, long-horizon robotic tasks are tackled using the “Sense-Plan-Act” (SPA) pipeline [Marton, 1984; Paul, 1981; Murphy, 2019], which involves perceiving the environment, planning tasks based on a dynamic model, and executing actions through low-level controllers. A common approach to implementing this pipeline involves using visual and language encoders to extract task-relevant features for representation learning, followed by training control policies with model-based visual reinforcement learning (RL) [Dalal et al., 2021; Yamada et al., 2021; Dalal et al., 2024]. While the above solutions are somewhat effective, they still face significant challenges in complex long-horizon tasks: (1) Language and visual encoders struggle to capture the hierarchical structure and dependencies of multi-stage sub-tasks in long-horizon tasks; and (2) Environmental feedback in such tasks is often sparse, while RL policies heavily relies on the rational reward structure. The former limits the robot’s ability to fully understand task dynamics and environmental context, while the latter further hinders the development of stable and effective long-horizon manipulation policies.

Our key insight is that achieving stable execution of long-horizon tasks in model-based visual RL relies on **enabling robots to accurately understand tasks, perceive multi-stage interactions between the robot and objects in the environment, and learn stable control policies through a structured reward system**. How can robots be equipped with these capabilities? We propose leveraging pre-trained large language models (LLMs) and visual demonstrations captured by multi-view cameras to empower robots, primarily because: 1) LLMs have made significant advancements in robotics, demonstrating capabilities such as step-by-step plan-

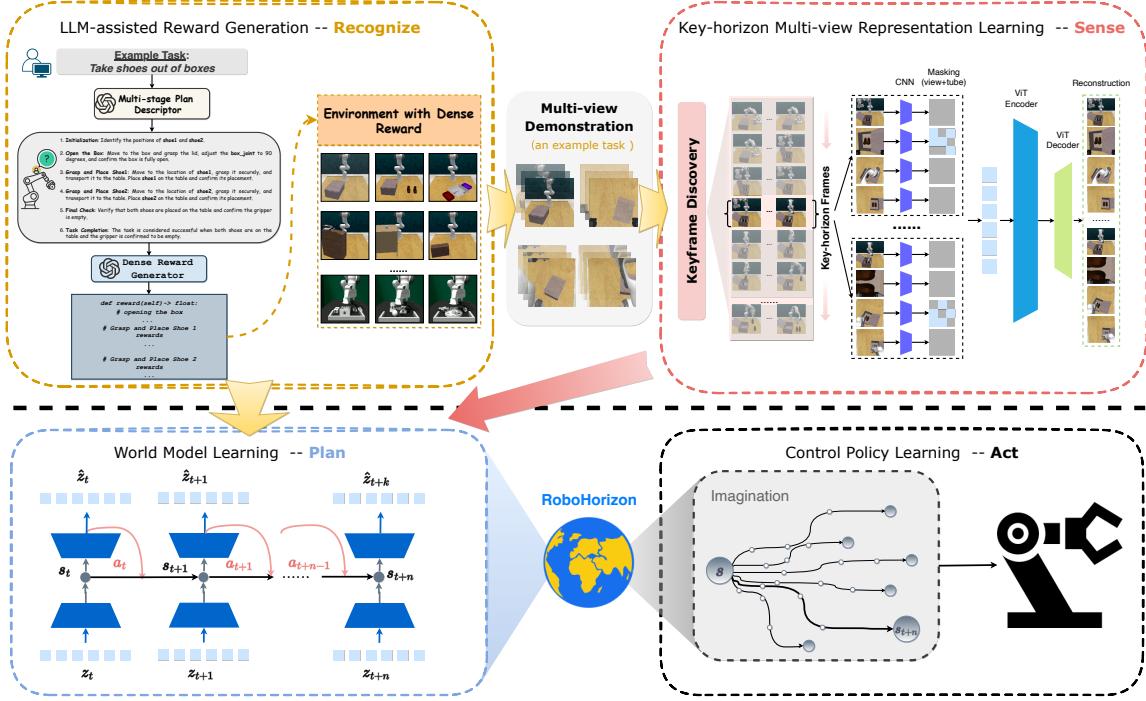


Figure 2: RoboHorizon overview, using the long-horizon robotic manipulation task “*take shoes out of box*” in RLBench as the illustration example, following the proposed **RSPA pipeline**.

ning [Liang *et al.*, 2023; Zeng *et al.*, 2022; Ahn *et al.*, 2022; Snell *et al.*, 2022], goal-oriented dialogue [Zeng *et al.*, 2022; Ahn *et al.*, 2022; Huang *et al.*, 2022], sub-goals [Huang *et al.*, 2023; Chen *et al.*, 2024a] and reward generation [Chiang *et al.*, 2019; Yu *et al.*, 2023] for robotic tasks based on language instructions. and 2) Observations from multi-camera views can significantly enhance a robot’s visual manipulation capabilities, and this setup is becoming increasingly common in real-world applications. Execution trajectories captured from different viewpoints often share similar environmental dynamics and physical structures. Previous studies have explored learning control policies from multi-view offline data using model-based RL [Seo *et al.*, 2023b] or imitation learning (IL) [Goyal *et al.*, 2023; Shridhar *et al.*, 2023; Ke *et al.*, 2024].

Building on the above insight, we extend the traditional SPA pipeline into the *Recognize-Sense-Plan-Act (RSPA)* pipeline, as illustrated in Fig. 1, specifically designed to address the challenges in long-horizon robotic manipulation. At the core of this pipeline is **RoboHorizon**, an LLM-assisted multi-view world model that enables robots to execute tasks effectively in complex, long-horizon robotic scenarios. To construct RoboHorizon, we first leverage pre-trained LLMs to generate a reasonable reward system for long-horizon tasks with sparse rewards. Unlike previous methods that use LLMs to generate reward signals directly applied to motion controllers [Chiang *et al.*, 2019; Yu *et al.*, 2023], which fail to address the complexities of multi-stage, long-horizon decision-making, our approach utilizes LLMs to divide long-horizon tasks into multi-stage sub-tasks. For each stage, dense stepwise rewards and intermediate rewards are generated, along with task reward code integrated into the en-

vironment interface, enabling robots to fundamentally **recognize** the long-horizon tasks they need to execute. Next, by collecting multi-view demonstrations of long-horizon manipulation tasks, we move beyond previous methods that directly perform representation learning on multi-view long-horizon demonstrations [Seo *et al.*, 2023b; Goyal *et al.*, 2023; Shridhar *et al.*, 2023]. Instead, we integrate the multi-view masked autoencoder (MAE) architecture [Seo *et al.*, 2023b; He *et al.*, 2021] with the keyframe discovery method in multi-stage sub-tasks [James and Davison, 2022], proposing a novel long-horizon key-horizon multi-view representation learning method. This method enables robots to more accurately **sense** interactions between the robotic gripper and objects during critical multi-task stages. Building on dense reward structures and the learned key-horizon multi-view representations, we construct RoboHorizon, enabling robots to effectively and efficiently **plan** action trajectories for long-horizon tasks. Finally, we leverage imagined trajectories generated by the world model to train RL policies, enabling effective low-level **act** capabilities for robots. We evaluate RoboHorizon on 4 short-horizon and 6 long-horizon tasks from the RLBench [James *et al.*, 2020] and 3 furniture assembly tasks from the FurnitureBench [Heo *et al.*, 2023], both representative testbeds for robotic manipulation under sparse rewards and multi-camera settings. RoboHorizon outperforms state-of-the-art model-based visual RL methods, with a 25.35% improvement on short-horizon RLBench tasks and 29.23% on long-horizon RLBench tasks and FurnitureBench assembly tasks.

Our contributions can be summarized as follows: **(1)** We introduce a novel *Recognize-Sense-Plan-Act (RSPA)* pipeline for long-horizon robotic manipulation, which tightly integrates LLMs for dense reward structures generation (*Recognize*),

key-horizon multi-view representation learning for interaction perceiving (Sense), a world model for action trajectories planning (Plan), and RL policies for robot control (Act). **(2)** Based on the *Recognize-Sense-Plan-Act (RSPA)* pipeline, we propose RoboHorizon, a robot world model specifically designed for long-horizon manipulation tasks, built upon LLM-generated dense reward structures and key-horizon multi-view representations. It enables efficient long-horizon task planning and ultimately ensures the stable execution of RL policies. **(3)** We provide a comprehensive empirical study of RoboHorizon’s performance in both short- and long-horizon manipulation tasks, validating RoboHorizon’s effectiveness enabled by the proposed *RSPA* pipeline.

2 Related Work

Methods for Long-horizon Manipulation Tasks Long-horizon robotic tasks are typically addressed through the “Sense-Plan-Act” (SPA) pipeline [Marton, 1984; Paul, 1981; Murphy, 2019]. This pipeline involves comprehensive environment perception, task planning based on dynamic models of the environment, and action execution via low-level controllers. Traditional methods encompass a range of techniques, from operation planning [Taylor *et al.*, 1987], grasp analysis [Miller and Allen, 2004] to task and motion planning (TAMP) [Garrett *et al.*, 2021] and skill-chaining [Chen *et al.*, 2024b]. Recent approaches, on the other hand, integrate vision-driven learning techniques [Mahler *et al.*, 2016; Sundermeyer *et al.*, 2021]. These algorithms enable long-horizon decision-making in complex, high-dimensional action spaces [Dalal *et al.*, 2024]. However, they often face challenges in handling contact-rich interactions [Mason, 2001; Whitney, 2004], are prone to cascading errors arising from imperfect state estimation [Kaelbling and Lozano-Pérez, 2013], and require extensive manual engineering [Garrett *et al.*, 2020]. Our work leverages pre-trained large language models (LLMs) for task recognition, extending the traditional SPA pipeline into the *Recognize-Sense-Plan-Act (RSPA)* pipeline, thereby significantly reducing the dependence on manual engineering. At the same time, our key-horizon multi-view representation learning method enhances the robot’s ability to perceive contact-rich interactions. Together, these innovations effectively address cascading failures between sub-tasks, enabling robust long-horizon planning using the developed world model.

Visual Robotic Control with Multi-View Observation Building on the latest advancements in computer vision and robotics learning, numerous methods have been developed to leverage multi-view data from cameras for visual control [Akinola *et al.*, 2020; Chen *et al.*, 2021; Hsu *et al.*, 2022; Chen *et al.*, 2023; Shridhar *et al.*, 2023; Seo *et al.*, 2023b]. Some of these methods utilize self-supervised learning to obtain view-invariant representations [Sermanet *et al.*, 2018], learn 3D keypoints [Chen *et al.*, 2021; Shridhar *et al.*, 2023; Ke *et al.*, 2024], or perform representation learning from different viewpoints [Seo *et al.*, 2023b] to address subsequent manipulation tasks. However, these approaches are often limited to short-horizon robotic visual control tasks and lack the ability to handle long-horizon, multi-view robotic visual representations.



Figure 3: Visualizing the RGB observations of keyframes from four camera viewpoints for the *take shoes out of the box* task using the keyframe discovery method, and displaying the key-horizon between the last two keyframes from the front viewpoint.

In contrast, our work aims to develop a framework that learns key-horizon representations for multi-stage sub-tasks from long-horizon, multi-view visual demonstrations, enabling robots to tackle various complex long-horizon visual control tasks.

3 Method

In this section, we detail the construction process of RoboHorizon, an LLM-assisted multi-view world model designed to achieve stable long-horizon robotic manipulation. First, we define the problem setting for the long-horizon manipulation tasks targeted in this work (Sec. 3.1). Next, we describe the LLM-assisted reward generation process (Recognize – Sec. 3.2) and key-horizon multi-view representation learning method (Sense – Sec. 3.3). Finally, we explain the development of the RoboHorizon world model (Plan – Sec. 3.4) and the implementation of robot control through RL policies (Act – Sec. 3.5).

3.1 Problem Setup

We consider a long-horizon task as a Partially Observed Markov Decision Process (POMDP) [Sutton *et al.*, 1999] defined by $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_0, \mathcal{O}, p_O, \gamma)$. Here, \mathcal{S} represents the set of environment states, \mathcal{A} the set of actions, $\mathcal{T}(s'|s, a)$ the transition probability distribution, $\mathcal{R}(s, a, s')$ the reward function, p_0 the initial state distribution, \mathcal{O} the set of observations, $p_O(O|s)$ the observation distribution, and γ the discount factor. A sub-task ω is a smaller POMDP $(\mathcal{S}, \mathcal{A}_\omega, \mathcal{T}, \mathcal{R}_\omega, p_0^\omega)$ derived from the full task’s POMDP. For example, in the task “*take shoes out of the box*”, the first sub-task is “*open the box*”. The next sub-task, “*grasp and place shoe 1*”, can only begin after the first is completed. When multiple sub-tasks are highly sequentially dependent, this forms the long-horizon tasks we focus on. In our case, the observation space consists of all RGB images. The reward function is generated by large language models (LLMs), and the task description is provided to the agent in natural language. We also assume the availability of multi-view demonstration data for the task: $\zeta_n^v = \{o_0^v, \dots, o_n^v\}$, where $v \in \mathcal{V}$ represents available viewpoints, and n is the total time step of the demonstrations.

3.2 LLM-assisted Reward Generation – Recognize

We present the RoboHorizon overview in Fig. 1, using the long-horizon manipulation task “*take shoes out of box*” in RLBench as the illustration example. Specifically, given a task’s language description, we prompt the LLMs to generate a corresponding task plan and encode dense rewards that align closely with each stage phase of the task. Following Yu *et al.* (2023), we decompose the process of translating language into rewards into two stages: multi-stage tasks description and dense reward generation. It is worth noting that while we reference the architecture of Yu *et al.* (2023), the internal prompts and tasks setting are entirely different.

As can be seen from the **Recognize** part in Fig. 1, in the Stage 1, we employ a pre-trained LLM as the Multi-stage Plan Descriptor that interprets and expands user input into detailed language descriptions of the required robot motions using predefined templates. To enable the multi-stage plan descriptor to generate a coherent structure for long-horizon tasks, we create a prompt template that outlines the current robot task setting. This leverages the pre-trained LLM’s internal knowledge of motion planning to produce detailed motion descriptions. In the Stage 2, we deploy another LLM as the Dense Reward Generator to convert these motion descriptions into corresponding reward functions. We approach this as a coding task, leveraging the pre-trained LLM’s understanding of coding and code structure. Four types of prompts guide the dense reward generator in generating the reward codes: i) task stage descriptions based on the task environment interface, ii) examples of expected reward generator responses, iii) constraints and rules for the reward encoder, and iv) specific task descriptions. Due to space limitations, example prompts for Stage 1 and Stage 2 of the *take shoes out of box* task are in the Appendix. Additionally, while any pre-trained language model can be used for reward generation, we find that only GPT-4o (OpenAI, 2024) reliably generates correct plans and rewards for all tasks. A detailed data flow of LLM-assisted reward generation is in the Appendix.

3.3 Key-horizon Multi-view Representation Learning – Sense

To enable robots to learn multi-stage interaction representations from long-horizon multi-view visual demonstrations, we propose the Key-Horizon Multi-View Masked Autoencoder (KMV-MAE) based on the MV-MAE architecture [Seo *et al.*, 2023b]. As shown in the **Sense** part of Fig. 1, our KMV-MAE method extracts key-horizons from multi-view demonstrations using the keyframe discovery method [James and Davison, 2022]. We then perform view-masked training on these key-horizons and use a video masked autoencoder to reconstruct missing pixels from masked viewpoints. Following prior work [Seo *et al.*, 2023a; Seo *et al.*, 2023b], we mask convolutional features instead of pixel patches and predict rewards to capture fine-grained details essential for long-horizon visual control.

Keyframe Discovery. The keyframe discovery method in our KMV-MAE, following previous works [James and Davison, 2022; Goyal *et al.*, 2023; Shridhar *et al.*, 2023; Ke *et al.*, 2024], identifies keyframes based on near-zero joint

velocities and unchanged gripper states. As illustrated in Fig. 3, this method captures each viewpoint’s keyframe $\mathcal{K}^v = \{k_1^v, k_2^v, \dots, k_m^v\}$ in the *take shoes out of the box* task from the demonstration ζ^v , where k represents the keyframe number. The corresponding time steps in the demonstration for each keyframe are $\{t_{k_1}, \dots, t_{k_m}\}$. Each adjacent keyframe pair k_i^v and k_{i+1}^v then forms a key-horizon $h_i = \{o_{t_{k_i}}^v, \dots, o_{t_{k_{i+1}}}^v\}$. Notably, the number of RGB observations in each key-horizon varies, depending on the time step difference between the adjacent keyframes in the demonstration.

View&Tube Masking and Reconstruction. To extract more interaction information from multi-view long-horizon demonstrations, we propose a view&tube masking method. For each frame, we randomly mask all features from three of the four viewpoints, while the remaining viewpoint has 95% of its patches randomly masked. Across the key-horizon, the unmasked viewpoint follows the tube masking strategy [Tong *et al.*, 2022]. This approach enhances cross-view feature learning, accounts for temporal correlations within a single viewpoint, reduces information leakage, and improves temporal feature representation. We integrate video masked autoencoding [Feichtenhofer *et al.*, 2022; Tong *et al.*, 2022] with the view&tube masking operation. Vision Transformer (ViT) [Dosovitskiy *et al.*, 2020] layers encode unmasked feature sequences across all viewpoints and frames. Following Seo *et al.* (2023a; 2023b), we concatenate mask tokens with the encoded features and add learnable parameters for each viewpoint and frame to align features with mask tokens. Finally, ViT layers decode the features, projecting them to reconstruct pixel patches while also predicting rewards to encode task-relevant information. This representation learning process can be summarized as:

Given demonstration videos $\zeta_n^v = \{o_0^v, \dots, o_n^v\}$, after m keyframes $\{k_1^v, k_2^v, \dots, k_m^v\}$ are extracted by the keyframe discovery method, they become in the form of containing $m - 1$ key-horizon: $\zeta^v = \{h_1^v, \dots, h_{m-1}^v\}_{v \in \mathcal{V}}$ from multiple viewpoints. Given LLM-assisted generated rewards $r = \{r_1, \dots, r_n\}$, and a mask ratio of m , KMV-MAE consists of following components:

$$\begin{aligned} \text{Convolution stem: } & l_i^v = f_\phi^{\text{conv}}(h_i^v) \\ \text{View&Tube masking: } & l_i^m \sim p^{\text{mask}}(l_i^m | \{h_i^v\}_{v \in \mathcal{V}}, m) \\ \text{ViT encoder: } & z_i^m \sim p_\phi(z_i^m | l_i^m) \\ \text{ViT decoder: } & \left\{ \begin{array}{l} \{\hat{h}_i^v\}_{v \in \mathcal{V}} \sim p_\phi(\{\hat{h}_i^v\}_{v \in \mathcal{V}} | z_i^m) \\ \hat{r}_{t_{k_i}, t_{k_{i+1}}} \sim p_\phi(\hat{r}_{t_{k_i}, t_{k_{i+1}}} | z_i^m) \end{array} \right. \end{aligned} \quad (1)$$

Finally the model is trained to reconstruct key-horizon pixels and predict rewards, i.e., minimizing the negative log-likelihood to optimize the model parameter ϕ as follows:

$$\mathcal{L}^{\text{kmvmae}}(\phi) = -\ln p_\phi(\{h_i^v\}_{v \in \mathcal{V}} | z_i^m) - \ln p_\phi(r_{t_{k_i}, t_{k_{i+1}}} | z_i^m)$$

3.4 RoboHorizon World Model – Plan

For the **Plan** part in Fig. 1, we construct RoboHorizon following previous works [Seo *et al.*, 2023a; Seo *et al.*, 2023b], implementing it as a variant of the Recurrent State Space Model (RSSM) [Hafner *et al.*, 2019]. The model uses frozen autoencoder representations from the previous key-horizon

multi-view representation learning as inputs and reconstruction targets. RoboHorizon includes the following components:

$$\begin{aligned} \text{Encoder: } & s_t \sim q_\theta(s_t | s_{t-1}, a_{t-1}, z_t) \\ \text{Decoder: } & \begin{cases} \hat{z}_t \sim p_\theta(\hat{z}_t | s_t) \\ \hat{r}_t \sim p_\theta(\hat{r}_t | s_t) \end{cases} \quad (2) \\ \text{Dynamics model: } & \hat{s}_t \sim p_\theta(\hat{s}_t | s_{t-1}, a_{t-1}) \end{aligned}$$

The encoder extracts state s_t from the previous state s_{t-1} , previous action a_{t-1} , and current autoencoder representations z_t . The dynamics model predicts s_t without access to z_t , allowing forward prediction. The decoder reconstructs z_t to provide learning signals for model states and predicts r_t to compute rewards from future states without decoding future autoencoder representations. All model parameters θ are optimized jointly by minimizing the negative variational lower bound [Kingma and Welling, 2014].:

$$\begin{aligned} \mathcal{L}^{\text{wm}}(\theta) = & -\ln p_\theta(z_t | s_t) - \ln p_\theta(r_t | s_t) \\ & + \beta \text{KL}[q_\theta(s_t | s_{t-1}, a_{t-1}, z_t) \| p_\theta(\hat{s}_t | s_{t-1}, a_{t-1})], \end{aligned}$$

where β is a scale hyperparameter.

3.5 Control Policy Learning – Act

For the **Act** part in Fig. 1, we build on the approach of [Seo et al., 2023a; Seo et al., 2023b] and adopt the actor-critic framework from DreamerV2 [Hafner et al., 2021]. The goal is to train a policy that maximizes predicted future values by backpropagating gradients through the RoboHorizon world model. Specifically, we define a stochastic actor and a deterministic critic as:

$$\text{Actor: } \hat{a}_t \sim p_\psi(\hat{a}_t | \hat{s}_t) \quad \text{Critic: } v_\xi(\hat{s}_t) \approx \mathbb{E}_{p_\theta} \left[\sum_{i \leq t} \gamma^{i-t} \hat{r}_i \right]$$

Here, the sequence $\{(\hat{s}_t, \hat{a}_t, \hat{r}_t)\}_{t=1}^H$ is predicted from the initial state \hat{s}_0 using the stochastic actor and dynamics model from Eq. 2. Unlike previous work, we set H to match the length of each key-horizon in the long-horizon task, with each key-horizon sequence having a different duration. Given the λ -return [Schulman et al., 2015] defined as:

$$V_t^\lambda \doteq \hat{r}_t + \gamma \begin{cases} (1-\lambda)v_\xi(\hat{s}_{t+1}) + \lambda V_{t+1}^\lambda & \text{if } t < H \\ v_\xi(\hat{s}_H) & \text{if } t = H \end{cases}$$

the critic is trained to regress the λ -return, while the actor is trained to maximize the λ -return with gradients backpropagated through the world model. To enable the robot to execute long-horizon tasks more reliably, we introduce an auxiliary behavior cloning loss that encourages the agent to learn expert actions while interacting with the environment. To achieve this, we follow the setup of [James and Davison, 2022; Seo et al., 2023b] to acquire the demonstration. Specifically, at each time step, given an expert action a_t^e , the objective of auxiliary behavior cloning is $\mathcal{L}^{\text{BC}} = -\ln p_\psi(a_t^e | s_t)$. Thus, the objective for the actor network and the critic network is:

$$\begin{aligned} \mathcal{L}^{\text{critic}}(\xi) & \doteq \mathbb{E}_{p_\theta, p_\psi} \left[\sum_{t=1}^{H-1} \frac{1}{2} (v_\xi(\hat{s}_t) - \text{sg}(V_t^\lambda))^2 \right] \\ \mathcal{L}^{\text{actor}}(\psi) & \doteq \mathbb{E}_{p_\theta, p_\psi} [-V_t^\lambda - \eta \text{H}[a_t | \hat{s}_t]] + \mathcal{L}^{\text{BC}} \end{aligned}$$

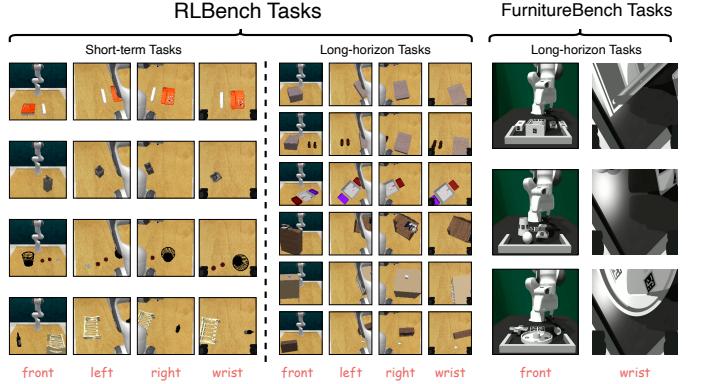


Figure 4: Visualization of multi-view demonstrations from front, left, right, and wrist cameras for 10 RLBench tasks, and from front and wrist cameras for 3 FurnitureBench tasks.

where sg is a stop gradient operation, η is a scale hyperparameter for an entropy $\text{H}[a_t | \hat{s}_t]$. Thus, with the generated dense reward structure, the training objective for the **sense**, **plan**, and **act** processes in RoboHorizon is to minimize the following objective function:

$$\mathcal{L}^{\text{RoboHorizon}} = \underbrace{\mathcal{L}^{\text{kmvmae}}}_{\text{Sense}} + \underbrace{\mathcal{L}^{\text{wm}}}_{\text{Plan}} + \underbrace{\mathcal{L}^{\text{critic}} + \mathcal{L}^{\text{actor}}}_{\text{Act}} \quad (3)$$

4 Experiments

We design our experiments to explore the following questions: (i) How does RoboHorizon, following the *RSPA* pipeline, perform compared to SPA-driven model-based reinforcement learning (RL) baselines in short- and long-horizon manipulation tasks? (ii) If SPA-based baseline methods also adopt stepwise rewards generated by LLMs but do not aware the staged reward for each sub-task stage, does RoboHorizon still maintain its advantage? (iii) To what extent do RoboHorizon’s key design components impact its overall performance?

Environmental Setup For quantitative evaluation, we adopt a demonstration-driven RL setup to address visual robotic manipulation tasks in RLBench [James et al., 2020] and FurnitureBench [Heo et al., 2023]. In both benchmarks, we rely on limited environment interactions and expert demonstrations. All experiments use only RGB observations from each camera, without incorporating proprioceptive state or depth information. Following previous studies [James and Davison, 2022; Seo et al., 2023b], we populate the replay buffer with expert demonstrations, and the RL agent outputs relative changes in the gripper’s position. For all tasks, 50 expert demonstrations are provided for each camera view. For FurnitureBench, we use a low-randomness environment initialization setup. Due to space limitations, more detailed experimental setups are provided in the Appendix.

Multi-view Camera Setup We adopt a multi-view observation and single-view control approach [Seo et al., 2023b], suitable for scenarios where multiple cameras are available during training, but the robot relies on a single camera during deployment. For RLBench tasks, we use multi-view data from front, wrist, left, and right cameras to enhance the robot’s

Table 1: Success Rates (%) on 4 short-horizon tasks (in RLBench) and 9 long-horizon tasks (6 in RLBench, 3 in FurnitureBench). Results are averaged over 5 seeds.

Model	Short-Horizon Tasks				Long-Horizon Tasks (RLBench)						FurnitureBench			Average	
	Phone on Base	Take Umbrella	Put Rubbish in Bin	Stack Wine	Take Shoes	Put Shoes	Empty Container	Put Books	Put Item	Slide Cabinet & Place Cups	Cabinet	Lamp	Round Table	Short Avg.	Long Avg.
TCN+WM	5.2	4.1	2.4	2.2	0	0	0	0.4	0	0	1.0	6.5	8.2	3.48	1.79
CLIP+WM	13.3	15.7	12.2	11.8	0	0	0	2.2	0	0	3.8	11.7	15.5	13.25	3.69
MAE+WM	20.4	19.1	18.6	19.6	0	0	0	2.5	0	0	8.5	16.3	20.9	19.43	5.36
MWM	32.5	30.8	28.4	29.7	1.2	0.8	2.1	3.3	1.1	0.5	15.2	27.5	32.0	30.35	9.30
MV-MWM	52.6	50.3	48.9	49.1	3.5	2.7	4.6	10.4	5.2	2.9	26.6	43.5	46.7	50.23	16.23
RoboHorizon	78.4	75.2	74.8	73.9	36.5	31.2	40.5	58.4	48.2	33.6	41.0	58.5	61.3	75.58 (25.35% ↑)	45.47 (29.23% ↑)

perception of long-horizon tasks and the environment, while training a RL agent that operates solely on front camera input. For FurnitureBench tasks, we use multi-view data from front and wrist cameras with the same training and control setup. We conduct experiments on 10 representative tasks in RLBench, including 4 short-horizon tasks (*phone on base*, *take umbrella out of stand*, *put rubbish in bin*, *stack wine*) and 6 long-horizon tasks (*take shoes out of box*, *put shoes in box*, *empty container*, *put books on bookshelf*, *put item in drawer*, *slide cabinet open and place cups*), as well as 3 long-horizon furniture assembly tasks in FurnitureBench (*cabinet*, *lamp*, *round table*), as shown in Fig. 4. In these tasks, the front, left, right cameras provide a wide view of the robot’s workspace, while the wrist camera offers close-up views of the target objects.

Baselines To compare with SPA-driven model-based RL methods using manually defined rewards, we select MV-MWM [Seo et al., 2023b] and MWM [Seo et al., 2023a] as baselines. The former lacks key-horizon representation learning, while the latter lacks multi-view key-horizon representation learning. Both baselines rely on manually defined rewards and the same amount of training data. Additionally, we adopt various representation learning methods to train world models, further demonstrating the effectiveness of our key-horizon multi-view representation learning in long-horizon tasks. The comparison methods include CLIP+WM [Radford et al., 2021], MAE+WM [He et al., 2021], and TCN+WM [Sermanet et al., 2018]. Specifically, RoboHorizon, MV-MWM, MWM, and TCN+WM learn representations from scratch, whereas CLIP+WM and MAE+WM use frozen pre-trained representations. More details about the experimental baselines are provided in the Appendix.

4.1 Performance Comparison

In this section, we conduct experiments on 4 short-horizon and 6 long-horizon tasks from RLBench, as well as 3 furniture assembly tasks from FurnitureBench, to address the three initial questions.

SPA-driven model-based RL baselines vs. RoboHorizon
 Table 1 compares the success rates of RoboHorizon and five SPA-driven baselines on 4 short-horizon tasks (in RLBench) and 9 long-horizon tasks (6 in RLBench, 3 in FurnitureBench). RoboHorizon outperforms all baseline methods, achieving the highest average success rate across all tasks. Specifically, it exceeds MV-MWM by 25.35% on the 4 short-horizon tasks

and by 29.23% on the 9 long-horizon tasks. This result shows that with LLM-assisted reward structures and key-horizon multi-view representation learning, RoboHorizon excels in both short-horizon pick-and-place tasks and long-horizon tasks with multiple sub-tasks and numerous target objects, achieving more stable manipulation. While MV-MWM benefits from multi-view representation learning, outperforming MWM, MAE+WM, CLIP+WM, and TCN+WM in representation, it still struggles with long-horizon tasks. Additionally, the manually designed reward structures are not robust, leading to inconsistent performance across all tasks for MV-MWM, MWM, MAE+WM, CLIP+WM, and TCN+WM.

These experimental results effectively answer our first question: **RoboHorizon outperforms SPA-driven baselines with manually defined rewards in both short- and long-horizon tasks, with a particularly strong advantage in long-horizon scenarios.** This outcome strongly validates the capability of our proposed RSPA pipeline in handling long-horizon tasks.

SPA-driven baselines with LLM-generated stepwise rewards vs. RoboHorizon To compare the performance of SPA-driven baselines with LLM-generated stepwise rewards but without staged rewards for each sub-task phase against RoboHorizon, we replace the manually defined reward system in SPA-driven baselines with the stepwise rewards generated during RoboHorizon’s Recognize phase using LLM assistance. Fig. 5 shows the success rate comparison between RoboHorizon and five SPA-driven baselines enhanced with LLM-generated stepwise rewards (as indicated by “LLM” in the legend) across 4 short-horizon tasks and 9 long-horizon tasks. It also illustrates the performance improvement of SPA-driven baselines with LLM-generated stepwise rewards compared to manually defined rewards. The results show that while these baseline methods achieve some improvement in task success rates when using LLM-generated stepwise rewards, demonstrating the effectiveness of well-designed stepwise rewards aligned with motion planning in helping robots understand operational tasks, their performance gains in long-horizon tasks remain very limited without considering staged rewards for each sub-task phase. Furthermore, regardless of the task, they fail to surpass the full RoboHorizon framework driven by our proposed RSPA pipeline.

This result clearly answers our second question: **By using LLM-generated stepwise rewards aligned with motion planning, SPA-driven baseline methods achieve certain performance improvements in both short-horizon and**

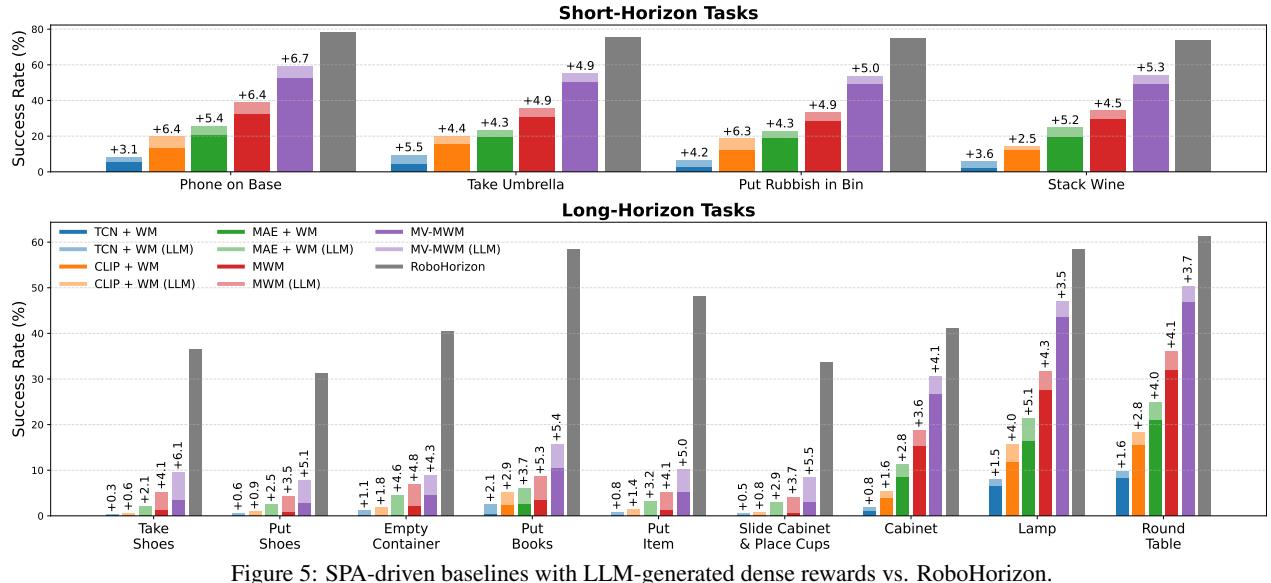


Figure 5: SPA-driven baselines with LLM-generated dense rewards vs. RoboHorizon.

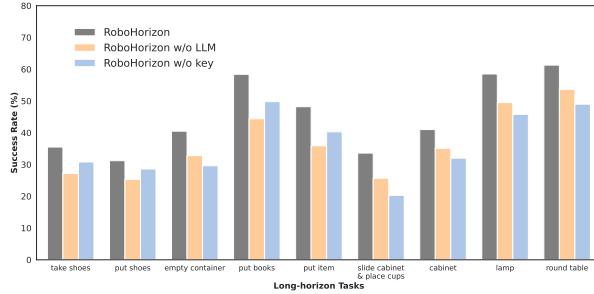


Figure 6: Ablation of key designs in RoboHorizon across 9 long-horizon tasks.

long-horizon tasks. However, due to their lack of consideration for staged rewards in each sub-task phase and their limited ability to learn complex long-horizon task representations, RoboHorizon still maintains a significant performance advantage. This finding further highlights the importance of the LLM-assisted reward generation mechanism and key-horizon multi-view representation learning modules in the RSPA-driven RoboHorizon framework. These modules not only enhance the robot’s ability to interpret task instructions but also improve its perception of tasks and target objects in complex long-horizon scenarios.

Ablation Study of RoboHorizon To evaluate the impact of RoboHorizon’s two key designs, LLM-assisted reward generation (recognition) and key-horizon multi-view representation learning (perception), on performance, we design two comparison methods: RoboHorizon w/o LLM, which removes LLM-assisted reward generation and relies on manually designed reward structures, and RoboHorizon w/o key, which omits key-horizon multi-view representation learning and uses the MV-MWM method for world model construction. Fig. 6 shows the ablation study results across nine long-horizon tasks. The results demonstrate that both LLM-assisted reward generation and key-horizon multi-view representation learning are critical to RoboHorizon’s performance, with each excelling in different scenarios. In tasks such as taking shoes out of a box, putting shoes in a box, placing books on a bookshelf, and

putting items in a drawer, where objects are dispersed and representation learning is easier, RoboHorizon w/o key performs better. In contrast, in tasks like emptying a container, sliding a cabinet open and placing cups, and installing a cabinet, lamp, and round table, where objects are densely packed with more distractions, RoboHorizon w/o LLM performs better.

These findings clearly answer the third key question: **Both LLM-assisted reward generation and key-horizon multi-view representation learning are indispensable for RoboHorizon’s overall performance. In tasks with dispersed objects, LLM-assisted reward generation plays a more significant role, whereas in tasks with densely distributed objects, key-horizon multi-view representation learning is more crucial.**

5 Conclusion and Discussion

In this paper, we propose a novel *Recognize-Sense-Plan-Act (RSPA)* pipeline for long-horizon manipulation tasks. The *RSPA* pipeline decomposes complex long-horizon tasks into four stages: recognizing tasks, sensing the environment, planning actions, and executing them effectively. Based on this pipeline, we develop **RoboHorizon**, an LLM-assisted multi-view world model. RoboHorizon uses its LLM-assisted reward generation module for accurate task recognition and its key-horizon multi-view representation learning module for comprehensive environment perception. These components enable RoboHorizon to build a robust world model that supports stable task planning and effective action execution through reinforcement learning algorithms. Experiments on RLBench and FurnitureBench show that RoboHorizon significantly outperforms state-of-the-art baselines in long-horizon tasks. Future work will focus on two key areas: enhancing the LLM-assisted reward generation in RoboHorizon by incorporating human feedback to create a closed-loop process that strengthens the framework’s adaptability to multiple tasks, and applying the RSPA pipeline and RoboHorizon model to real-world long-horizon robotic manipulation tasks to improve the framework’s sim-to-real transfer capabilities.

References

- [Ahn *et al.*, 2022] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 2
- [Akinola *et al.*, 2020] Iretiayo Akinola, Jacob Varley, and Dmitry Kalashnikov. Learning precise 3d manipulation from multiple uncalibrated cameras. In *ICRA*, pages 4616–4622. IEEE, 2020. 3
- [Chen *et al.*, 2021] Boyuan Chen, Pieter Abbeel, and Deepak Pathak. Unsupervised learning of visual 3d keypoints for control. In *ICLR*, pages 1539–1549. PMLR, 2021. 3
- [Chen *et al.*, 2023] Zixuan Chen, Wenbin Li, Yang Gao, and Yiyu Chen. Tild: Third-person imitation learning by estimating domain cognitive differences of visual demonstrations. In *AAMAS*, pages 2421–2423, 2023. 3
- [Chen *et al.*, 2024a] Yangtao Chen, Zixuan Chen, Junhui Yin, Jing Huo, Pinzhuo Tian, Jieqi Shi, and Yang Gao. Gravmad: Grounded spatial value maps guided action diffusion for generalized 3d manipulation. *arXiv preprint arXiv:2409.20154*, 2024. 2
- [Chen *et al.*, 2024b] Zixuan Chen, Ze Ji, Jing Huo, and Yang Gao. Scar: Refining skill chaining for long-horizon robotic manipulation via dual regularization. In *NeurIPS*, 2024. 3
- [Chiang *et al.*, 2019] Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2):2007–2014, 2019. 2
- [Dalal *et al.*, 2021] Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *NeurIPS*, 34:21847–21859, 2021. 1
- [Dalal *et al.*, 2024] Murtaza Dalal, Tarun Chiruvolu, Devendra Chaplot, and Ruslan Salakhutdinov. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. *arXiv preprint arXiv:2405.01534*, 2024. 1, 3
- [Dosovitskiy *et al.*, 2020] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 4
- [Feichtenhofer *et al.*, 2022] Christoph Feichtenhofer, Yang-hao Li, Kaiming He, et al. Masked autoencoders as spatiotemporal learners. *NeurIPS*, 35:35946–35958, 2022. 4
- [Garrett *et al.*, 2020] Caelan Reed Garrett, Chris Paxton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. Online replanning in belief space for partially observable task and motion problems. In *ICRA*, pages 5678–5684. IEEE, 2020. 3
- [Garrett *et al.*, 2021] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021. 3
- [Goyal *et al.*, 2023] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *CoRL*, pages 694–710. PMLR, 2023. 2, 4
- [Hafner *et al.*, 2019] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICLR*, pages 2555–2565. PMLR, 2019. 4
- [Hafner *et al.*, 2021] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. 5
- [He *et al.*, 2021] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 2, 6
- [Heo *et al.*, 2023] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023. 2, 5, 13
- [Hsu *et al.*, 2022] Kyle Hsu, Moo Jin Kim, Rafael Rafailov, Jiajun Wu, and Chelsea Finn. Vision-based manipulators need to also see from their hands. *arXiv preprint arXiv:2203.12677*, 2022. 3
- [Huang *et al.*, 2022] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *ICLR*, pages 9118–9147. PMLR, 2022. 2
- [Huang *et al.*, 2023] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *CoRL*, pages 540–562. PMLR, 2023. 2
- [James and Davison, 2022] Stephen James and Andrew J Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation. *IEEE Robotics and Automation Letters*, 7(2):1612–1619, 2022. 2, 4, 5
- [James *et al.*, 2020] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. 2, 5, 13
- [Kaelbling and Lozano-Pérez, 2013] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013. 3
- [Ke *et al.*, 2024] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024. 2, 3, 4

- [Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014. 5
- [Liang *et al.*, 2023] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, pages 9493–9500. IEEE, 2023. 2
- [Mahler *et al.*, 2016] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *ICRA*, pages 1957–1964. IEEE, 2016. 3
- [Marton, 1984] J. Marton. Scientific fundamentals of robotics 1. dynamics of manipulation robots: Theory and application: Edited by miomir vukobratovic and veljko potkonjak. *Autom.*, 20(2):265–266, 1984. 1, 3
- [Mason, 2001] Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001. 3
- [Miller and Allen, 2004] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. 3
- [Murphy, 2019] Robin R Murphy. *Introduction to AI robotics*. MIT press, 2019. 1, 3
- [Paul, 1981] Richard P Paul. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981. 1, 3
- [Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICLR*, 2021. 6
- [Schulman *et al.*, 2015] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 5
- [Seo *et al.*, 2023a] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *CoRL*, pages 1332–1344. PMLR, 2023. 4, 5, 6
- [Seo *et al.*, 2023b] Younggyo Seo, Junsu Kim, Stephen James, Kimin Lee, Jinwoo Shin, and Pieter Abbeel. Multi-view masked world models for visual robotic manipulation. In *ICLR*, pages 30613–30632. PMLR, 2023. 2, 3, 4, 5, 6, 13
- [Sermanet *et al.*, 2018] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018. 3, 6
- [Shridhar *et al.*, 2023] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, pages 785–799. PMLR, 2023. 2, 3, 4
- [Snell *et al.*, 2022] Charlie Snell, Mengjiao Yang, Justin Fu, Yi Su, and Sergey Levine. Context-aware language modeling for goal-oriented dialogue systems. *arXiv preprint arXiv:2204.10198*, 2022. 2
- [Sundermeyer *et al.*, 2021] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-grasnet: Efficient 6-dof grasp generation in cluttered scenes. In *ICRA*, pages 13438–13444. IEEE, 2021. 3
- [Sutton *et al.*, 1999] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *NeurIPS*, 12, 1999. 3
- [Taylor *et al.*, 1987] Russ H Taylor, Matthew T Mason, and Kenneth Y Goldberg. Sensor-based manipulation planning as a game with nature. In *Fourth International Symposium on Robotics Research*, pages 421–429, 1987. 3
- [Tong *et al.*, 2022] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *NeurIPS*, 35:10078–10093, 2022. 4
- [Whitney, 2004] Daniel E Whitney. *Mechanical assemblies: their design, manufacture, and role in product development*, volume 1. Oxford university press New York, 2004. 3
- [Yamada *et al.*, 2021] Jun Yamada, Youngwoon Lee, Gautam Salhotra, Karl Pertsch, Max Pflueger, Gaurav Sukhatme, Joseph Lim, and Peter Englert. Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. In *CoRL*, pages 589–603. PMLR, 2021. 1
- [Yu *et al.*, 2023] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasen- clever, Jan Humplík, et al. Language to rewards for robotic skill synthesis. In *CoRL*, pages 374–404. PMLR, 2023. 2, 4
- [Zeng *et al.*, 2022] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022. 2

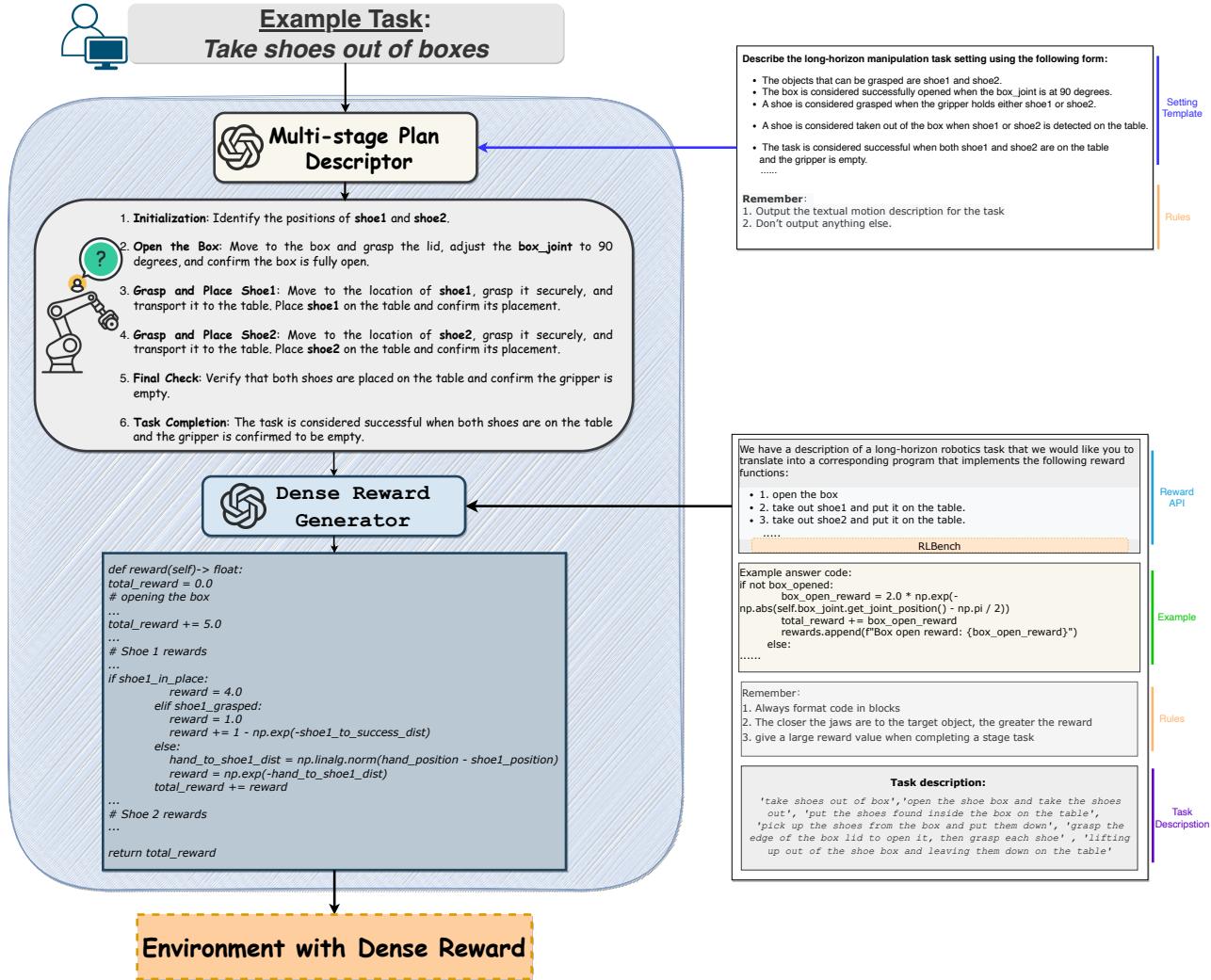


Figure 7: Detailed data flow of the LLM-assisted Reward Generation, the *take shoes out of box* task as the illustration example.

A Technical Appendix

In this technical appendix, we first provide a detailed example illustration of the data flow in the LLM-assisted Reward Generation (the **Recognize** part) and the two-stage (**Multi-stage Plan Descriptor** and **Dense Reward Generator**) prompts examples. Then, we present a detailed introduction to the RLBench and FurnitureBench benchmark used in the experiments, along with the related tasks.

A.1 Detailed LLM-Assisted Reward Generation

As shown in Fig. 7, the LLM-assisted Reward Generation process is divided into two stages. We use the long-horizon task “*take shoes out of the box*” from RLBench as a demonstration to illustrate the two stages.

In Stage 1, we employ a pre-trained LLM as the **Multi-stage Plan Descriptor** that interprets and expands user input into detailed language descriptions of the required robot motions using predefined templates. To enable the multi-stage plan descriptor to generate a coherent structure for long-horizon tasks, we create a prompt template that outlines the current robot task setting. This leverages the pre-trained LLM’s internal knowledge of motion planning to produce detailed motion descriptions.

In Stage 2, we introduce another LLM as the **Dense Reward Generator**, which translates motion descriptions into corresponding dense reward structures. Specifically, for each sub-task stage, the model generates smaller **stepwise rewards** (e.g., rewards for each action during the grasping of shoe1) and larger **staged rewards** (e.g., rewards for successfully grasping shoe1). This process is treated as a coding task, leveraging the pre-trained LLM’s deep understanding of code and its structure. The Dense Reward Generator is guided by four types of prompts to generate reward code: i) task stage descriptions based on the task environment interface, ii) examples of expected outputs from the reward generator, iii) constraints and rules for the reward encoder, and iv) specific task descriptions.

We provide a detailed two-stage data flow for the “*take shoes out of the box*” task, along with the related prompts, the plans generated in Stage 1 and the reward codes generated in Stage 2.

Data Flow for Stage 1

Prompt: Describe the long-horizon manipulation task “*take shoes out of the box*” using the following format:

- Graspable objects are shoe1 and shoe2.
- The box is considered successfully opened when the box_joint is at 90 degrees.
- A shoe is considered grasped when the gripper holds either shoe1 or shoe2.
- A shoe is considered out of the box when it is detected on the table.
- The task is considered successful when both shoe1 and shoe2 are placed on the table.

Remember to:

- Provide only the motion description for the task.
- Exclude any other information.

Plan: [Motion Description for the Task]

1. **Initialize:** Identify shoe1 and shoe2.
2. **Open the Box:** Move towards the box and adjust the box_joint to 90 degrees. Confirm the box is open.
3. **Grasp and Place Shoe1:** Move towards shoe1, grasp it, move to the table, and place shoe1 on the table. Confirm shoe1 is on the table.
4. **Grasp and Place Shoe2:** Move towards shoe2, grasp it, move to the table, and place shoe2 on the table. Confirm shoe2 is on the table.
5. **Final Check:** Ensure both shoes are on the table and the gripper is empty.
6. **Task Completion:** The task is successful when both shoes are placed on the table and the gripper is empty.

Data Flow for Stage 2

Prompt: We have a description of a long-horizon robotics task “*take shoes out of the box*” that we would like you to translate into a corresponding code that implements the following reward functions:

- Open the Box: Move towards the box and adjust the box joint to 90 degrees. Confirm the box is open.
- Grasp and Place Shoe1: Move towards shoe1, grasp it, move to the table, and place shoe1 on the table. Confirm shoe1 is on the table.
- Grasp and Place Shoe2: Move towards shoe2, grasp it, move to the table, and place shoe2 on the table. Confirm shoe2 is on the table.
- Final Check: Ensure both shoes are on the table and the gripper is empty.
- Task Completion: The task is successful when both shoes are placed on the table and the gripper is empty.

Example answer code:

```
if not box_opened:  
    box_open_reward = 2.0 * np.exp(-np.abs(self.box_joint.get_joint_position() - np.pi / 2))  
    total_reward += box_open_reward  
    rewards.append(f"Box open reward: {box_open_reward}")  
else:  
    total_reward += 5.0 # Box opened reward  
    rewards.append("Box opened reward: 5.0")
```

Remember:

- Always format code in blocks
- The closer the jaws are to the target object, the greater the reward
- Give a large reward value when completing a stage task

Task description:

‘take shoes out of box’, ‘open the shoe box and take the shoes out’, ‘put the shoes found inside the box on the table’, ‘pick up the shoes from the box and put them down’, ‘grasp the edge of the box lid to open it, then grasp each shoe’, ‘lift them up out of the shoe box and leave them down on the table’

Generated Rewards:

```
def reward(self) -> float:  
    rewards = []  
    total_reward = 0.0  
  
    box_opened = self._opened_cond_box.condition_met()[0]  
  
    # Reward for opening the box  
    if not box_opened:  
        box_open_reward = 2.0 * np.exp(-np.abs(self.box_joint.get_joint_position() - np.pi / 2))  
        total_reward += box_open_reward  
        rewards.append(f"Box open reward: {box_open_reward}")  
    else:  
        total_reward += 5.0 # Box opened reward  
        rewards.append("Box opened reward: 5.0")  
  
    # Shoe 1 rewards  
    shoe1_grasped = self._grasped_cond_shoe1.condition_met()[0]  
    shoe1_in_place = self._detected_cond_shoe1.condition_met()[0]  
    shoe1_position = self.shoe1.get_position()  
    hand_position = self.robot.arm.get_tip().get_position()  
    shoe1_to_success_dist = np.linalg.norm(shoe1_position - self.success_sensor.get_position())  
  
    if shoe1_in_place:  
        reward = 4.0  
        rewards.append(f"Shoe1 in place reward: 4.0")  
    elif shoe1_grasped:  
        reward = 1.0  
        rewards.append(f"Shoe1 grasped reward: 1.0")  
        reward += 1 - np.exp(-shoe1_to_success_dist)  
        rewards.append(f"Shoe1 moving towards target reward: {1 - np.exp(-shoe1_to_success_dist)}")  
    else:  
        hand_to_shoe1_dist = np.linalg.norm(hand_position - shoe1_position)  
        reward = np.exp(-hand_to_shoe1_dist)  
        rewards.append(f"Shoe1 approach reward: {np.exp(-hand_to_shoe1_dist)}")  
  
    total_reward += reward  
  
    # Shoe 2 rewards  
    shoe2_grasped = self._grasped_cond_shoe2.condition_met()[0]  
    shoe2_in_place = self._detected_cond_shoe2.condition_met()[0]  
    shoe2_position = self.shoe2.get_position()  
    shoe2_to_success_dist = np.linalg.norm(shoe2_position - self.success_sensor.get_position())  
  
    if shoe2_in_place:  
        reward = 4.0  
        rewards.append(f"Shoe2 in place reward: 4.0")  
    elif shoe2_grasped:  
        reward = 1.0  
        rewards.append(f"Shoe2 grasped reward: 1.0")  
        reward += 1 - np.exp(-shoe2_to_success_dist)  
        rewards.append(f"Shoe2 moving towards target reward: {1 - np.exp(-shoe2_to_success_dist)}")  
    else:  
        hand_to_shoe2_dist = np.linalg.norm(hand_position - shoe2_position)  
        reward = np.exp(-hand_to_shoe2_dist)  
        rewards.append(f"Shoe2 approach reward: {np.exp(-hand_to_shoe2_dist)}")  
  
    total_reward += reward  
  
    return total_reward
```

A.2 Experimental Details

Simulation environment We use the RLBench [James *et al.*, 2020] and FurnitureBench [Heo *et al.*, 2023] simulator. In the RLBench environment, we conduct experiments using a 7-DoF Franka Panda robot arm equipped with a parallel gripper on 4 short-horizon and 6 long-horizon visual manipulation tasks. In the FurnitureBench environment, we perform experiments with the same robot configuration on 3 long-horizon furniture assembly tasks.

Data collection To achieve key-horizon multi-view representation learning and policy learning that combines reinforcement learning with behavior cloning, we first collect expert data across both types of simulation tasks. For demonstration data collection in RLBench, we double the maximum velocity of the Franka Panda robot arm in PyRep [?], which reduces the duration of the demonstrations without significantly compromising their quality. For each short-horizon task, we use RLBench’s dataset generator to collect 50 demonstration trajectories per camera view, and for each long-horizon task, we collect 100 demonstration trajectories per camera view. For data collection in the FurnitureBench tasks, we utilize the automated furniture assembly scripts provided by the platform to automate the data collection process. Similarly, for each long-horizon furniture assembly task, we collect 100 demonstration trajectories per camera view.

Implementation Our implementation is built on the official MV-MWM [Seo *et al.*, 2023b] framework, and unless otherwise specified, the implementation details remain the same. To expedite training and mitigate the bottleneck caused by a slow simulator, we run 8 parallel simulators. Our autoencoder is composed of an 8-layer ViT encoder and a 6-layer ViT decoder, with an embedding dimension set to 256. We maintain a consistent set of hyperparameters across all experiments.

Computing hardware For all RLBench experiments, we use a single NVIDIA GeForce RTX 4090 GPU with 24GB VRAM and it takes 12 hours for training MV-RoboWM and 16 hours for training MV-MWM.

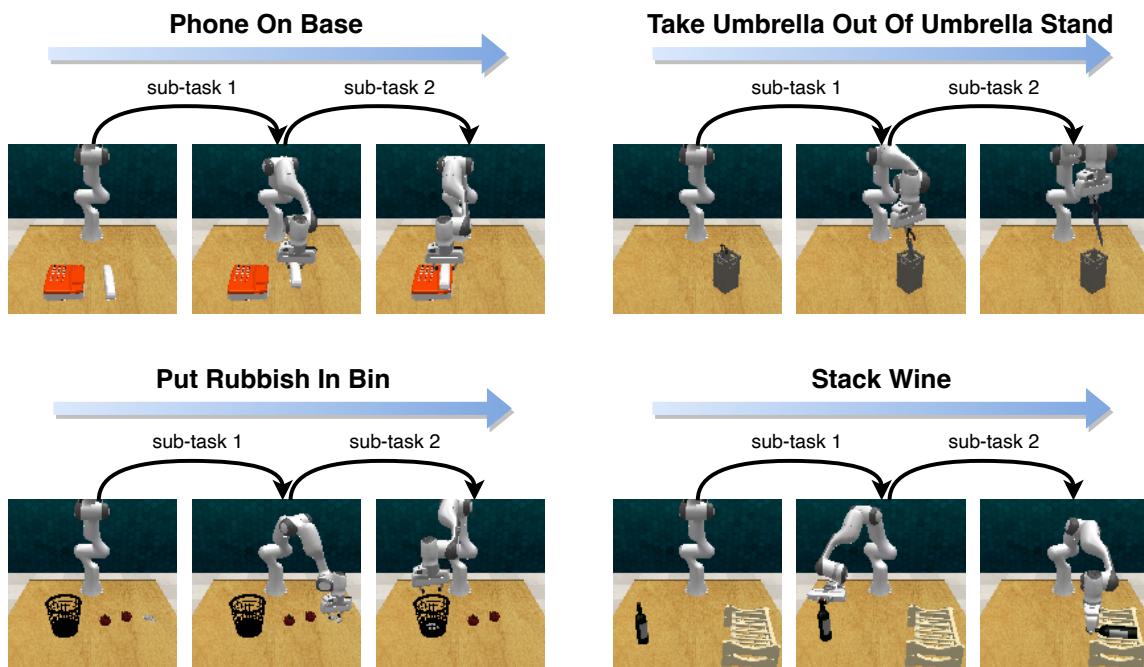


Figure 8: Visualization of the 4 short-horizon RLBench tasks in the experiment.

A.3 RLBench Tasks

We select 10 tasks from the 100 available in RLBench for our simulation experiments, including 4 short-horizon tasks (as shown in Fig. 8) and 6 long-horizon tasks (as shown in Fig. 9). To reduce training time with limited resources, we only use variation0. In the following sections, we describe each of these 10 tasks in detail, including any modifications made to the original codebase.

Phone On Base

Task: grasp the phone and put it on the base.

filename: `phone_on_base.py`

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: The phone is on the base.

Task Horizon: 2.

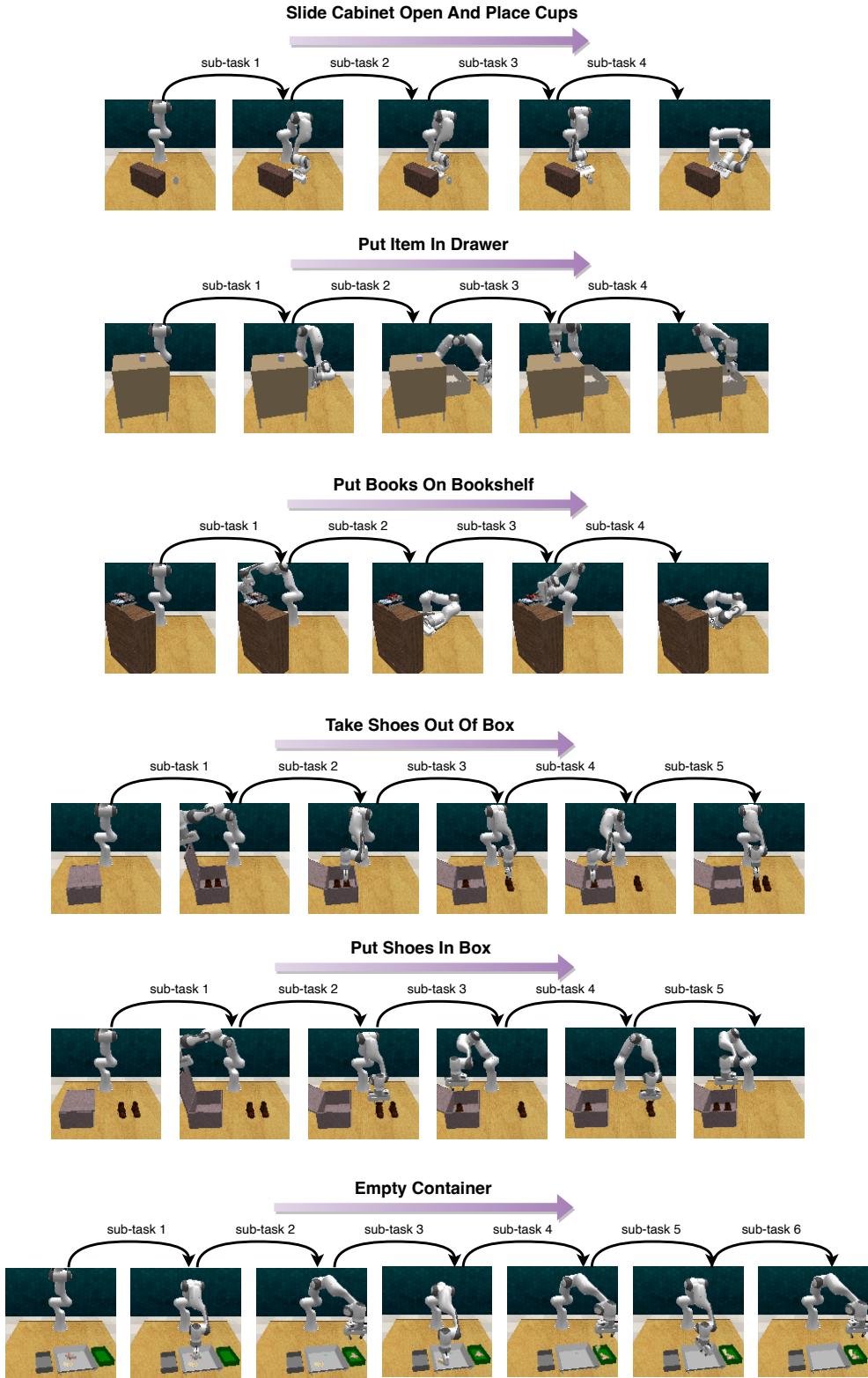


Figure 9: Visualization of the 6 long-horizon RLBench tasks in the experiment.

Take Umbrella Out Of Umbrella Stand

Task: grasp the umbrella by its handle, lift it up and out of the stand.

filename: take_umbrella_out_of_umbrella_stand.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: The umbrella is taken off the stand.

Task Horizon: 2.

Put Rubbish In Bin

Task: pick up the rubbish and leave it in the trash can.

filename: put_rubbish_in_bin.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: The rubbish is thrown in the bin.

Task Horizon: 2.

Stack Wine

Task: place the wine bottle on the wine rack.

filename: stack_wine.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: The bottle is on the wine rack.

Task Horizon: 2.

Take Shoes Out Of Box

Task: grasp the edge of the box lid to open it, then grasp each shoe, lifting up out of the shoe box and leaving them down on the table.

filename: take_shoes_out_of_box.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: Both shoes are placed on the table.

Task Horizon: 5.

Put Shoes In Box

Task: open the box lid and put the shoes inside.

filename: put_shoes_in_box.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: Both shoes are placed in the box.

Task Horizon: 5.

Empty Container

Task: move all objects from the large container and drop them into the smaller red one.

filename: empty_container.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: All objects in the large container are placed in the small red container.

Task Horizon: 6.

Put Books On Bookshelf

Task: pick up 2 books and place them on the top shelf.

filename: put_books_on_bookshelf.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: All the books are placed on top of the shelf.

Task Horizon: 4.

Put Item In Drawer

Task: open the middle drawer and place the block inside of it.

filename: put_item_in_drawer.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: The block is placed in the middle drawer.

Task Horizon: 4.

Slide Cabinet Open And Place Cups

Task: grasping the left handle, open the cabinet, then pick up the cup and set it down inside the cabinet.

filename: slide_cabinet_open_and_place_cups.py

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: The cup is in the left cabinet.

Task Horizon: 5.

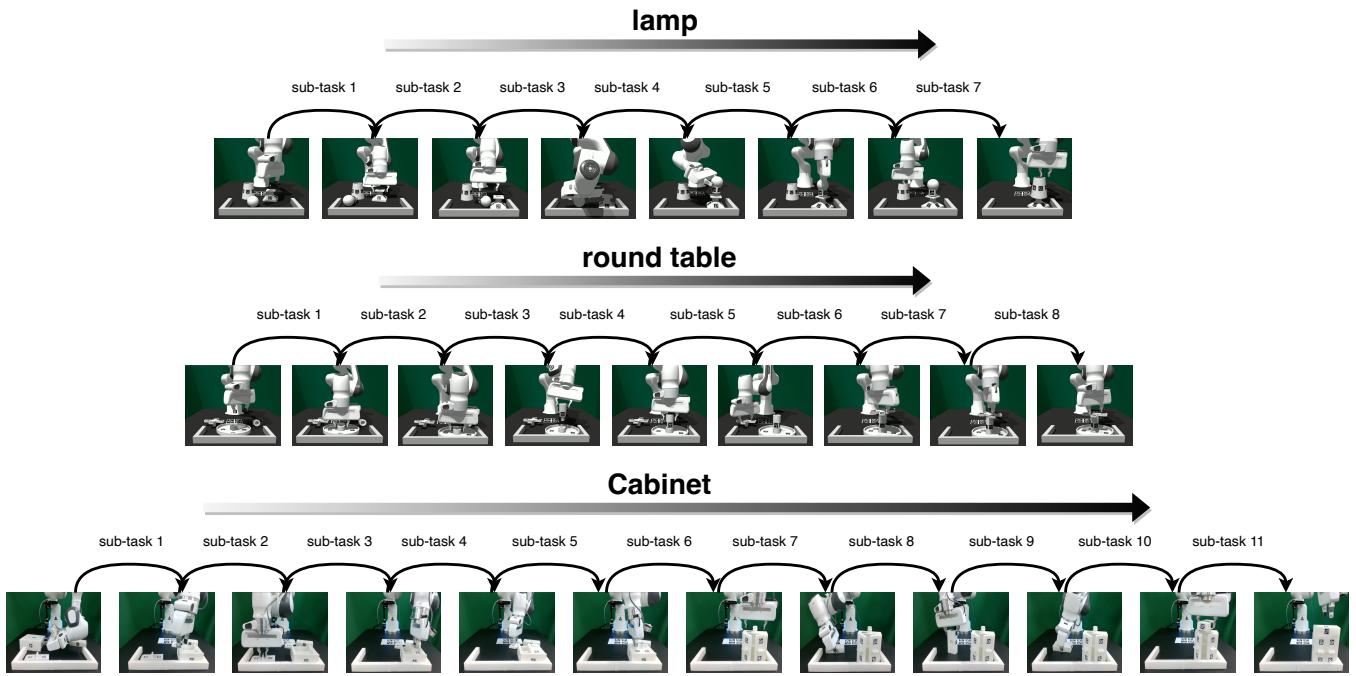


Figure 10: Visualization of the 3 long-horizon FurnitureBench tasks in the experiment, where the assembly of the cabinet is illustrated as the process with a real robot.

A.4 FurnitureBench Tasks

We select 3 furniture assembly tasks from the 9 available task in FurnitureBench for our experiments, as shown in Fig. 10). In the following parts, we describe each of these 3 tasks in detail, including any modifications made to the original codebase.

Lamp

Task: The robot needs to screw in a light bulb and then place a lamp hood on top of the bulb. The robot should perform sophisticated grasping since the bulb can be easily slipped when grasping and screwing due to the rounded shape

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: Three pieces of furniture are assembled into a lamp.

Task Horizon: 7.

Round Table

Task: The robot should assemble one rounded tabletop, rounded leg, and cross-shaped table base. The robot should handle an easily movable round leg and cross-shaped table base, which requires finding a careful grasping point.

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: Three pieces of furniture are assembled into an round table.

Task Horizon: 8.

Cabinet

Task: The robot must first insert two doors into the poles on each side of the body. Next, it must lock the top, so the doors do not slide out. This task requires careful control to align the doors and slide into the pole. Moreover, diverse skills such as flipping the cabinet body, and screwing the top are also needed to accomplish the task

Modified: Rewards are defined for each step based on the LLM-assisted reward generation module.

Success Metric: Four pieces of furniture are assembled into a cabinet.

Task Horizon: 11.