ExploRLLM: Guiding Exploration in Reinforcement Learning with Large Language Models

Runyu Ma, Jelle Luijkx, Zlatan Ajanović, and Jens Kober

***Abstract*— In image-based robot manipulation tasks with large observation and action spaces, reinforcement learning struggles with low sample efficiency, slow training speed, and uncertain convergence. As an alternative, large pre-trained foundation models have shown promise in robotic manipulation, particularly in zero-shot and few-shot applications. However, using these models directly is unreliable due to limited reasoning capabilities and challenges in understanding physical and spatial contexts. This paper introduces ExploRLLM, a novel approach that leverages the inductive bias of foundation models (e.g. Large Language Models) to guide exploration in reinforcement learning. We also exploit these foundation models to reformulate the action and observation spaces to enhance the training efficiency in reinforcement learning. Our experiments demonstrate that guided exploration enables much quicker convergence than training without it. Additionally, we validate that ExploRLLM outperforms vanilla foundation model baselines and that the policy trained in simulation can be applied in real-world settings without additional training. Code and videos are available at https://explorllm.github.io**

I. INTRODUCTION

Foundation models (FMs) [1], which refer to models trained on large-scale data (e.g. Large Language Models or Vision-Language Models), have shown significant promise in robotics. Large Language Models (LLMs) such as GPT-4 [2] demonstrate the ability to generate human-like commonsense-aware reasoning in some scenarios. This reasoning ability has been demonstrated as a zero-shot planner [3], capable of breaking down complex tasks into detailed step-by-step plans without additional training. Our paper focuses on pick-and-place manipulation tasks, a domain where LLMs are recently employed to provide task-grounding for high-level planning, executed by pre-trained low-level robot skills [4]. Furthermore, when integrated with Vision-Language Models (VLMs), LLMs utilize cross-domain knowledge to achieve robot perception and planning in manipulation tasks [5]. This synergy between LLMs and VLMs is also harnessed to extract environmental affordances and constraints, forming a basis for subsequent robotic planning [6]. Despite the impressive results achieved using FMs, it is important to note that unpredictable failures of LLM predictions can still result in robotic errors, and their use does not always ensure success and they in general do not learn from past experiences [7], [8].

To address these issues, we propose to add residual reinforcement learning [9] to affordances recognized by FMs. Reinforcement Learning (RL), as described in [10], provides a powerful framework for learning decision-making and control policies for robotics [11] through interactions with the environment. Despite the inherent errors in the information provided by FMs, RL can adaptively learn to compensate for those errors through trial and update. A key challenge in deep RL is the "curse of dimensionality", where large observation and action spaces hinder the agent's ability to explore and converge efficiently. To overcome this challenge, we utilize an LLM and a VLM to create a more compact and effective observation space. Moreover, we introduce an object-centric residual action space, defining the pick-or-place actions as positional adjustments relative to the centers of detected objects.

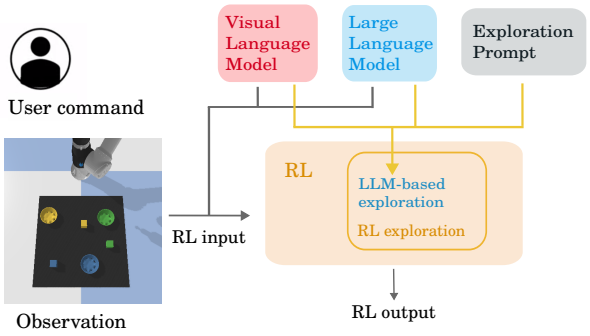


Fig. 1: Graphical illustration of the ExploRLLM framework.

Although actions generated by the LLM could be sub-optimal or sometimes lead to failures, we employ these actions to guide exploration in RL. Previous exploration strategies for RL (e.g., ϵ-greedy and Boltzmann exploration [10]) explored the state-action space in a stochastic manner, which focuses on the exploration-exploitation trade-off. However, these methods lack guided mechanisms as they do not incorporate prior knowledge to expedite convergence. Therefore, we utilize the LLM as a few-shot planner to create actions that act as exploration steps in RL. This strategy increases the likelihood of encountering successful states, thereby gathering more relevant state-action pairs for the off-policy RL agent.

Our method, ExploRLLM, enhances the robot system by incorporating reinforcement learning with FMs. This integration ensures enhanced performance compared to the plan generated by FMs, compensating for their inherent sub-optimality and biases. In turn, the FMs aid in accelerating RL training convergence by reducing the observation spaces and directing the exploration process. To summarize, our main contributions are:

The authors are with Cognitive Robotics, Delft University of Technology, The Netherlands (e-mail: {j.d.luijkx, z.ajanovic, j.kober}@tudelft.nl, r.ma-8@student.tudelft.nl).

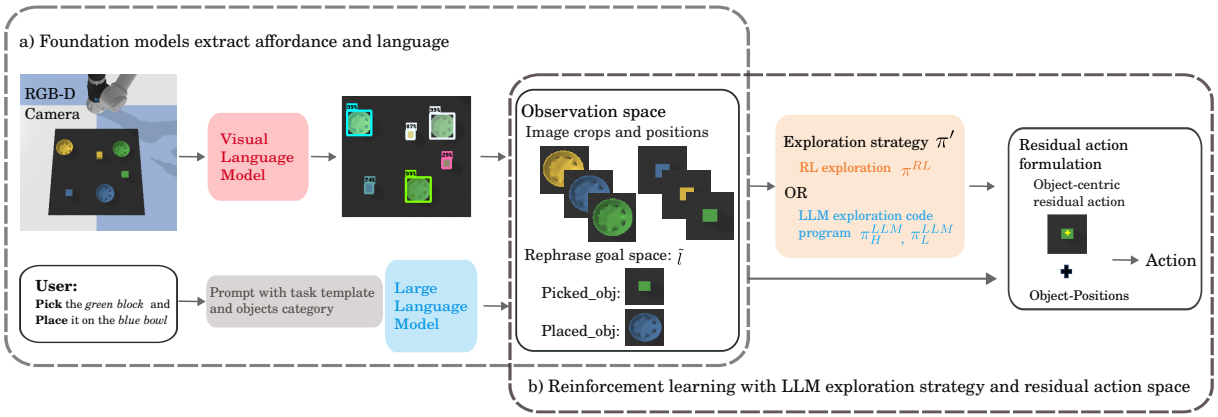


Fig. 2: Structure of ExploRLLM: a) The LLM reformulates user-provided language commands l with predefined templates and highlights the important objects of the template into a command vector $\tilde{l}$. In parallel, the VLM detects objects mentioned in the task and extracts crops centered at bounding box locations. b) RL takes the extracted image crops and object positions as input and uses the RL agent exploration method or LLM policy programs as exploration actions.

1) We propose ExploRLLM, which employs an RL agent with a) residual action and observation spaces derived from affordances identified by FMs and b) uses an LLM to guide exploration.
2) We develop the prompting method for LLM-based exploration using hierarchical-language-model-programs and demonstrate that our exploration method significantly shortens RL's convergence time.
3) We show that ExploRLLM achieves better final performance than the policies derived solely from the LLM and VLM by comparing it to several state-of-the-art baselines. We also show that the ExploRLLM policy can be transferred to unseen colors, letters, tasks, and real-world settings without additional training.

II. Related Work

Recently, there has been increasing interest in integrating robotics and the FMs by either developing robotics foundation models (e.g., RT-2 [12], PALM-E [13]) or applying pre-trained FMs to robotics. The application of FMs in robotics primarily falls into two categories: leveraging FMs for zero-shot or few-shot plan generation and using FMs to enhance RL training efficiency. Our work combines both aspects to solve robot manipulation tasks.

A. Foundation Models for Planning in Robotics

Researchers have demonstrated the ability of LLMs to create zero-shot or few-shot plans through reasoning capabilities [3], [14]. This ability is essential for devising high-level plans in the field of robotics. For instance, SayCan [4] utilizes LLMs to provide high-level planning in robot manipulation tasks. Additionally, InnerMonologue [15] showcased the potential of LLMs in processing feedback sources and interactively planning, facilitating closed-loop behaviors without additional training. Additionally, the code completion features of LLMs have been utilized in creating robot-centric formulations of LLM-generated programs for robot skills [16] and task plans [17].

VLMs have been increasingly integrated into robotics. For example, CLIPort [18] proposed an end-to-end imitation learning framework that leverages the broad semantic capabilities of CLIP [19] to interpret language instructions and visual inputs. Socratic Models [5] integrates LLMs with vision language model ViLD [20] to utilize cross-domain knowledge to achieve robot perception and planning for zero-shot manipulation tasks. VoxPoser [6] deployed a VLM and an LLM to compose a 3D value map in observation space for a model-based planning framework to zero-shot synthesize closed-loop robot trajectories. However, directly applying VLMs and LLMs for zero-shot tasks does not guarantee success and safety, as the physical properties of the real world remain challenging for those FMs. Instead, our work considers the actions generated by LLMs and VLMs to be exploratory behaviors within an RL framework.

B. Foundation Models and Reinforcement Learning

Incorporating FMs into RL frameworks has notably improved RL's effectiveness. In [21], the authors have implemented LLMs as proxy reward functions, demonstrating their utility in RL. In the context of RL for robotics, LLMs are also capable of generating reward signals for robot actions by connecting commonsense reasoning with low-level actions [22], self-refinement [23] and evolutionary optimization over reward code to enable complex tasks such as dexterous manipulation [24]. Regarding exploration, authors in [25] reward RL agents toward human-meaningful intermediate behaviors by prompting an LLM. LLMs are also utilized as an intrinsic reward generator to guide exploration for long horizon manipulation tasks [26]. Contrary to these studies, our approach directly employs LLM-generated code policies to guide exploratory actions rather than focusing on reward shaping.

III. Problem Formulation

This study aims to improve the effectiveness of RL agents engaged in robotic pick-and-place tasks. For such manipu-

lation tasks, each episode is initiated with a goal described in a linguistic term, represented by l. The agent, at every time step t, perceives an observation $\boldsymbol{o}^t$, consisting of an overhead RGB-D image and the state of the end effector. In other tabletop manipulation tasks (e.g., Transporter [27]), the action space is structured as a pick-and-place primitive, denoted as $\{\mathcal{P}_{\text{pick}}, \mathcal{P}_{\text{place}}\}$, where each action contains a position for pick and a position for place in top-down view coordinate. Our method, however, simplifies the action space to a single motion primitive: either pick or place. This simplification aims to make the RL challenge more tractable. The Pick-or-Place action primitive is defined as a tuple containing the primitive index $prim$ (0 for pick, 1 for place) and a top-down view position, expressed as $\mathcal{P}$, $\boldsymbol{a}^t = \{prim^t, \mathcal{P}^t\}$. At each time step, the agent receives a reward r from the environment comprising a dense reward component, denoted as r^d, and an external sparse reward component, referred to as r^s.

IV. Framework: ExploRLLM

Our method utilizes FMs to enhance RL training through the extraction of objects for observation spaces (Sec.IV-A), the creation of an object-centric action space (Sec.IV-B), and the direction of exploration based on guidance from LLMs (Sec.IV-C).

A. Observation Spaces based on Foundation Models

Our methodology leverages the strengths of LLMs and VLMs to extract the observation space used for the RL framework, as depicted in Figure 2. LLMs reformulate user-provided language commands into predefined templates and highlight the objects within these templates to form an interpreted command vector $\tilde{\boldsymbol{l}}$. For example, it identifies the "picked object" in a template like "put [picked object] on [placed object]". It is important to note that, within a given task setting, the number and category of objects do not change. Utilizing VLMs as open-vocabulary object detectors, our system identifies and encloses objects relevant to the task within bounding boxes from image in raw observation space $\boldsymbol{o}^t$, represented by their locations $\mathbf{P}_{\text{vlm}} = \{\mathcal{P}_{\text{vlm}_1}, \mathcal{P}_{\text{vlm}_2}, ...\}$. RGB-D visual inputs are segmented into crops based on bounding box positions, denoted as $\mathbf{I}_{\text{vlm}} = \{\mathcal{I}_{\text{vlm}_1}, \mathcal{I}_{\text{vlm}_2}, ...\}$. This method improves the system's robustness to detection inaccuracies and varying object shapes. The interpreted commands $\tilde{\boldsymbol{l}}$, the positional data $\mathbf{P}_{\text{vlm}}$ and the image patches $\mathbf{I}_{\text{vlm}}$ are then integrated into the reformulated RL observation $\boldsymbol{s}^t$.

B. Residual Action Spaces

As the VLM already extracts each object's position $\mathbf{P}^t_{\text{vlm}}[i^t]$, the action space is converted into an object-centric residual action space, as shown in Figure 2. The reformulated action space consists of a primitive index $prim$, an object index i and a residual position $\mathcal{P}_{\text{res}}$, expressed as $\tilde{\boldsymbol{a}}^t = \{prim^t, i^t, \mathcal{P}^t_{\text{res}}\}$, where $\mathcal{P}^t = \mathbf{P}^t_{\text{vlm}}[i^t] + \mathcal{P}^t_{\text{res}}$. For example, consider the task of picking the letter 'O', where $\mathbf{P}^t_{\text{vlm}}[i^t]$ denotes the center of the bounding box. In this case, a residual action $\mathcal{P}^t_{\text{res}}$ is needed to prevent failures due to the empty center of the object.

Algorithm 1 Exploration strategy π^{EXPL}

Input: state $\boldsymbol{s}^t$, instruction $\tilde{\boldsymbol{l}}$, LLM policies π_H^{LLM}, π_L^{LLM}
Parameter: threshold ϵ
Output: action $\tilde{\boldsymbol{a}}^t$

```
Sample a random number j from U(0, 1)
if j ≤ ε then
    Run LLM generated high level action policy π_H^LLM
        A^t = (prim^t, i^t) = π_H^LLM(s^t, l̃)
    Run LLM generated low level action policy π_L^LLM
        P^t_res = π_L^LLM(s^t, A^t)
        ã^t = (prim^t, i^t, P^t_res)
else
    Run the reinforcement learning policy π^RL
        ã^t = π^RL(s^t)
end if
return action ã^t
```

C. LLM-based Exploration in RL

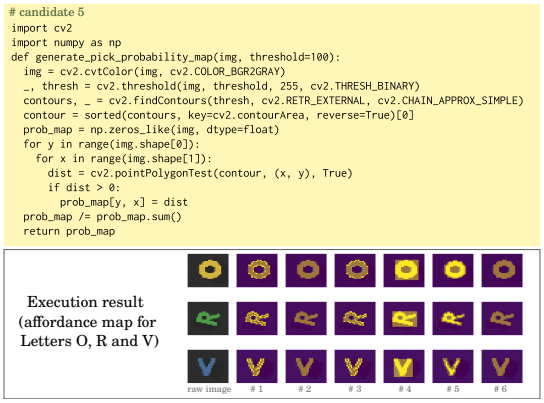
Traditional deep RL algorithms (e.g., SAC [28], PPO [29]) do not inherently ensure frequent visits to high-value states in high-dimensional state-action spaces, which becomes particularly challenging in vision-based tabletop manipulation tasks. In such cases, the RL agent may struggle to achieve favorable outcomes when successful results are infrequent. By utilizing the planning capabilities of LLMs and the perception capabilities of VLMs, we can leverage the rich prior knowledge within these FMs to direct the exploration process more effectively.

The LLM-based exploration strategy, denoted as π^{EXPL} in Algorithm 1, draws inspiration from the ϵ-greedy strategy. Specifically, during the rollout collection at each timestep, the off-policy RL agent employs the LLM-based exploration technique if a sampled random variable falls below the threshold ϵ. Otherwise, the action is selected according to the current RL agent's policy, π^{RL}, as detailed in Algorithm 1.

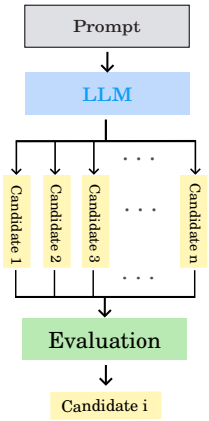
For the creation of plans in robotic manipulation tasks, prior research often prompts LLMs on every step to generate plans. However, this method of frequent LLM invocation during the training phase is highly resource-intensive, incurring significant time and financial costs due to the numerous iterations required to train a single RL agent. Code as Policy (CaP) [16] shows that LLMs are proficient at devising policy by generating robot-centric formulation programs. Drawing inspiration from CaP, our methodology employs the LLM to hierarchically generate language model programs, which are then executed iteratively during the training phase as exploratory actions, enhancing efficiency and resource utilization.

The hierarchical language model programs include both high-level π_H^{LLM} and low-level π_L^{LLM} policy code programs. A high-level plan primarily involves selecting robot action primitives and the objects to interact with based on the current state of the robot and the objects.

(a) A prompt example that combines the task description, VLM scene description, and image. It then generates code policy candidates for low-level skills.



(b) LLM-generated python code and affordance heatmaps for unseen objects.



(c) Evaluation phase

Fig. 3: The LLM generates low-level policy code.

In contrast to high-level tasks, instructing low-level actions poses a more significant challenge because high-level states and actions are more accessible and can be represented as language. When dealing with low-level actions, the complexity of the state becomes considerably more intricate, particularly for image-based problems. Therefore, instead of a deterministic code policy, we instruct the LLM to produce a code policy π_L^{LLM} for generating an affordance map according to the input image. The low-level exploration behavior is derived from a stochastic policy that relies on the values within this affordance map. Although the code generated by LLMs lacks guaranteed feasibility and accuracy in robot environments, these models can generate potentially useful policy candidates, with the one exhibiting the highest success rate being selected as shown in Figure 3c.

V. Implementation

The main components for implementation of ExploRLLM are RL agent, VLM-based object detection and code policy generation by LLM.

1) Reinforcement learning agent: We use the Soft Actor-Critic (SAC) algorithms with modifications in the collecting rollout phase, detailed in Algorithm 1. Other implementation aspects remain consistent with the standard SAC approach in stable-baselines3 [30]. Figure 4 illustrates the network architecture for RL. We employ two convolutional layers to transform every image patch into a vector $\boldsymbol{x} \in \mathbb{R}^{n \times d}$, where n is the number of objects captured by VLM and d signifies the dimension of each patch as encoded by the CNN. The vector is subsequently concatenated with the position, robot gripper state, and the extracted episodic language goal $\tilde{\boldsymbol{l}}$ to form a new vector $\boldsymbol{x'} \in \mathbb{R}^{n \times d'}$, where d' denotes the dimension of each patch's vector following encoding and concatenation. The self-attention layer is used to transform vector $\boldsymbol{x'}$. It is linearly transformed to query $\boldsymbol{Q} \in \mathbb{R}^{n \times d'}$, key $\boldsymbol{K} \in \mathbb{R}^{n \times d'}$ and values $\boldsymbol{V} \in \mathbb{R}^{n \times d'}$. Then the self-attention module is applied as: $\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}$. The output features from this layer then

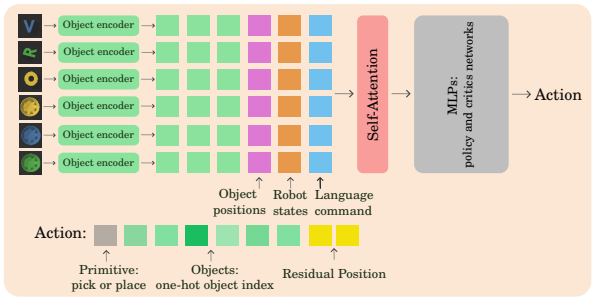


Fig. 4: ExploRLLM RL architecture and action space.

go into a two-layer MLP. The aforementioned structure is consistently utilized across all actor and critic networks.

2) VLM detection: Utilizing an open-vocabulary object detector ViLD [20], objects in the environment can be identified by given specific labels. However, implementing this model online during training is time-consuming, so ViLD is utilized solely in the evaluation phase. In the training phase, the ground truth in the simulation is used to determine the center positions of the bounding boxes. It is important to note that ViLD's position detection in real-world scenarios is not always flawless. To simulate this imperfection, noise following a Gaussian distribution with a standard deviation equivalent to half the radius of the image crop is applied to the ground truth positions.

3) Code policy generation by LLM: The policy code for executing high-level behavior is obtained using a few-shot prompt in GPT-4 [2]. This prompt includes a list of available robot motion primitives to demonstrate the robot's actions. A custom API is also provided to aid the LLM in reasoning, such as determining whether an object is held in the robot's gripper or understanding the relationships between different objects. Following the approach demonstrated by [16], where LLMs have been shown capable of generating novel policy codes with example codes and commands, our prompt also includes these examples. They are designed to guide the LLM in formulating plans and conducting geometric rea-

soning for our specific task scenarios.

For low-level exploration actions, we employ GPT4 with Vision [2], which generates code using prompts that combine example images with language descriptions, enriching the context with visual information, as shown in Figure 3. The provided example images include a depiction of the environmental setup featuring the robot, a simulated background, objects, and a specific example of image patches inside VLM bounding boxes. The prompt describes the requirements and guidelines, enabling generated code to create a probability affordance heatmap for the specified image patch, utilizing external libraries like OpenCV and NumPy.

However, as indicated in Figure 3b, there are instances where the generated affordance map may not be optimal. For example, the optimal pick position for the letter 'O' should be at its rim, whereas the heatmap suggests the center. To address those sub-optimum problems, we adopt a stochastic policy based on the affordance map rather than a deterministic policy that selects the point of highest affordance. Given that RL evolves through rewards obtained from environmental interactions, the sub-optimality in exploration policy can be identified and rectified through learning. This approach also offers the potential to provide counter-examples during the phase of replay buffer collection.

VI. Experimental Setups

For experiments, we used a simulation and real-world setup of different tabletop pick-and-place tasks.

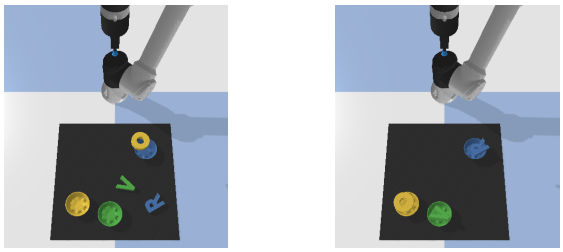
A. Simulation Setup

The proposed method is trained and evaluated in a simulated tabletop pick-and-place task, as depicted in Figure 5. Similar to [27] and [18], all simulated experiments are based on a Universal Robot UR5e with a suction gripper, and the input observation is a top-down RGB-D image.

Our task setting draws inspiration from the "Pick the [pick_color] box and place it in the [place_color] bowl" task described in [18]. Considering that picking at the center of the bounding box, as generated by the VLMs, is sufficiently accurate for a block, we have increased the challenge by substituting the blocks with various objects (such as letters). We assess our method across two tasks: a short-horizon (SH) task, "Pick the [pick_letter] and place it in the [place_color] bowl" and a long-horizon (LH) task, "Put all letters in the bowl of the corresponding color". In the short-horizon task, each episode begins with three letters and three bowls randomly placed on the table, with the objects for the pick-and-place action randomly chosen to create language commands. This task is completed only when the robot accurately places the selected letter in the specified bowl. For the long-horizon task, both letters and bowls are randomly arranged on the table at the beginning of each episode. This task is considered complete only when each letter is correctly placed in a bowl whose color matches the letters.

B. Real-world Setup

We validated our approach on a Franka-Emika Panda robot equipped with a Schmalz suction gripper and a RealSense D405 RGB-D camera, as shown in Figure 8a, implementing our policy and code in the EAGERx [31] framework.



(a) Short-horizon: Place [pick_letter] in the [place_color] bowl (b) Long-horizon: Put all letters in bowls of matching colors

Fig. 5: Simulation environment settings

Given the potential risks to hardware and the time-intensive nature of direct training, we completed training in simulation, with real-robot applications limited to evaluation. Object recognition used ViLD for bounding box identification based on object names. To simulate real-world conditions more accurately, we introduce noise to the bounding box center's position during the training phase in the simulation, mimicking the positional uncertainty inherent in VLM detection. We enhanced simulation realism by introducing noise to bounding box positions and image inputs, simulating VLM detection uncertainty and camera noise, including lighting variations.

VII. Results

We conducted a series of experiments to evaluate our approach. We assess the impact of LLM-guided exploration on training speed (SecVII-A), compare the success rate of ExploRLLM against foundation models alone (SecVII-B), and examine the generalization capabilities of our method in novel scenarios (SecVII-C). We also deployed ExploRLLM in the real-world setting without further training (SecVII-D).

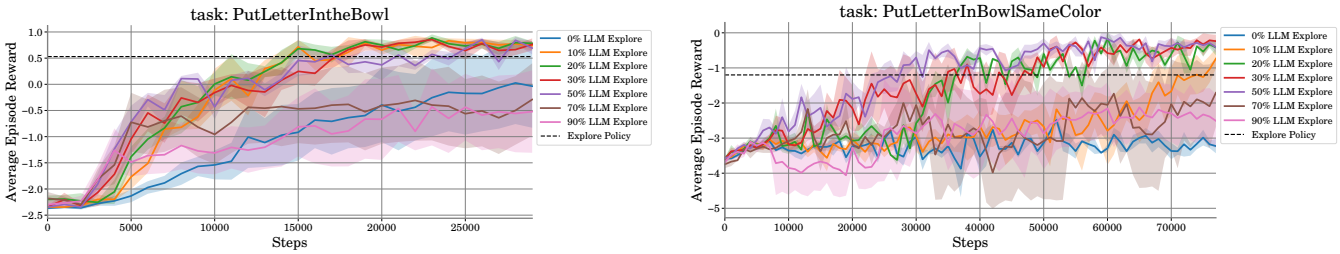
A. LLM-based Exploration Behavior Performance

In our experiment, we investigated how varying the frequency of LLM-based exploration affects training convergence, with $\epsilon \in \{0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 0.9\}$, as shown in Figure 6. An ϵ of 0 implies a pure Soft Actor-Critic method. We conducted training with six random seeds per frequency to evaluate the mean and variance in performance, starting each session with a 20,000-step warm-up phase without LLM exploration, given no significant policy improvements were observed in this initial phase. Post-warm-up results, depicted in Figure 6 and detailed in Table II for short and long-horizon tasks, demonstrate that ExploRLLM outperforms LLM-only policies across various exploration frequencies.

In the short-horizon task depicted in Figure 6a, the training process tends to be unstable without LLM-based exploration actions, leading to outcomes that can be both a successful policy or an unconverged one. When the exploration frequency is within $0 < \epsilon \leq 0.5$, the training becomes more stable and converges more swiftly, with minor variations across different ϵ settings. However, increasing ϵ beyond

TABLE I: Comparison of success and error rates of our method against baselines during 50 evaluation episodes with both seen and unseen colors. Evaluation includes different tasks: short-horizon (SH) and long-horizon (LH) and different initialization methods: no overlapping between objects (NO) and allowed overlapping (AO). For the ExploRLLM policy, the standard deviations of 6 seeds are included to show the stability of the training process.

Method	Overall success rate				Low-level error rate			
	SH NO	SH AO	LH NO	LH AO	SH NO	SH AO	LH NO	LH AO
ExploRLLM (20%)	0.86±0.05	0.80±0.06	0.70±0.11	0.54±0.09	0.14±0.05	0.20±0.06	0.18±0.10	0.22±0.9
ExploRLLM (0%)	0.56±0.40	0.48±0.36	–	–	0.32±0.24	0.42±0.30	–	–
CaP*	0.60	0.48	0.38	0.30	0.38	0.52	0.42	0.48
Socratic Models + CLIPort	0.78	0.64	0.50	0.36	0.22	0.28	0.22	0.28
Inner Monologue + CLIPort	0.82	0.72	0.58	0.42	0.18	0.26	0.20	0.24



(a) Short-horizon task: Pick the [pick letter] and place it in the [place color] bowl.

(b) Long-horizon task: Put all letters in the bowl of the corresponding color.

Fig. 6: Training curve with LLM-based exploration behavior

TABLE II: ExploRLLM training rewards with different ϵ

Explore ϵ (%)	SH Task (25k steps)	LH Task (75k steps)
0	-0.03 ± 1.13	-3.22 ± 0.29
10	0.74 ± 0.13	-0.73 ± 0.40
20	0.79 ± 0.06	-0.42 ± 0.31
30	0.76 ± 0.16	-0.23 ± 0.26
50	0.70 ± 0.17	-0.40 ± 0.23
70	-0.29 ± 0.98	-1.71 ± 1.38
90	-0.52 ± 1.12	-2.51 ± 1.09
Exploration Policy	0.53	-1.2

0.5 reduces the online data proportion, slowing progress and introducing higher instability into the training process.

For long-horizon tasks, Figure 6b illustrates a clear pattern where higher frequencies of LLM-based exploration, when $0 < \epsilon \leq 0.5$, correlate with faster training speeds. Those statistics demonstrate that LLM-based exploration is crucial for a relatively difficult task in obtaining experience close to the optimal region. This approach effectively mitigates issues associated with extensive observation and action spaces. However, akin to the findings in short-horizon tasks, a high rate of exploration leads to less stable training dynamics and slower convergence.

B. RL Performance

To evaluate the effectiveness of ExploRLLM, we benchmark its performance against four baselines: ExploRLLM without LLM-based exploration policy, CaP-style policy [16] (our exploration policy), Socratic Models [5], and Inner Monologue [15]. Our implementation of Socratic Models and Inner Monologue utilizes ViLD [20] as an object detector and GPT-4 [2] as a multi-step planner. The commands of individual steps are then executed by a pre-trained CLIPort [18] model with 500 demonstrations. The key difference between Socratic Models and Inner Monologue is that Inner Monologue features a success detector that can identify errors in the previous step.

During the evaluation phase, the spectrum of letter colors includes both seen and unseen colors. The evaluation encompasses a variety of tasks and initialization methods. "NO" indicates scenarios where there is no overlap between the positions of letters and bowls at the beginning of each episode, whereas "AO" allows overlaps. These configurations are designed to evaluate the robustness of each method when dealing with complex geometric relationships between objects.

For short-horizon tasks, as illustrated in Table I, ExploRLLM shows stable performance, unlike versions without the exploration policy, which sometimes fail to converge and display significant variance in success rates and low-level errors. Our method exceeds LLM-generated policies in success rate, minimizes robot behavior errors, and reduces

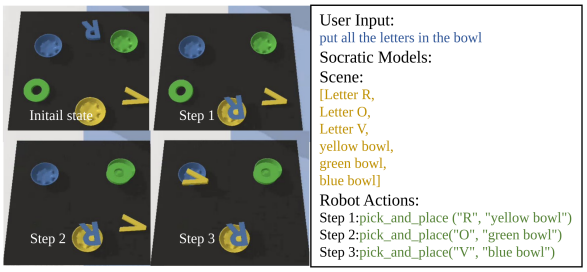
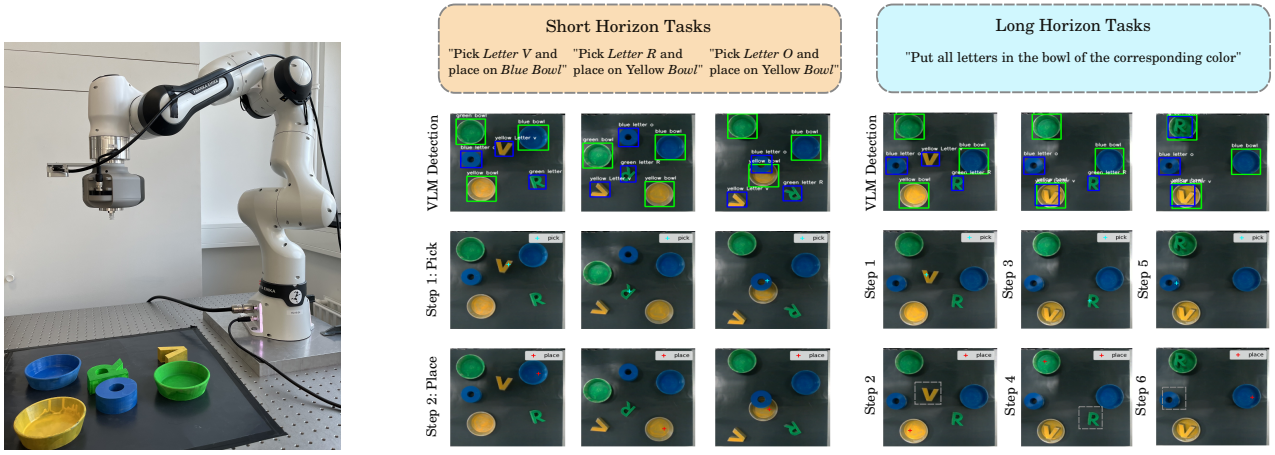


Fig. 7: Applying the single-step agent into zero-shot LLM planers (e.g., the Socratic Models).



(a) Real robot experiment settings

(b) Pick and Place position

Fig. 8: Real robot experiments demonstrate the practical application of ExploRLLM in real-world settings. Initially trained exclusively in a simulated environment, ExploRLLM, with the help of VLMs, effectively adapts to real-world scenarios. It can be deployed in both short-horizon and long-horizon tasks.

the gap between NO and AO scenarios, highlighting our exploration policy's effectiveness in correcting FMs' inaccuracies. In contrast, the CLIPort-based method struggles with novel scenarios or complex geometric relationships between objects. For long-horizon tasks, RL agents without LLM-based exploitation failed to converge. As depicted in Table I, ExploRLLM outperforms Socratic Models, Inner Monologue, and the policy generated by LLMs, showcasing superior performance for long-horizon tasks.

C. Generalization to Unseen Long-horizon Tasks

Despite our short-horizon agent being specifically trained for a pre-defined pick-and-place task, our approach maintains the capability to transfer from short-horizon policy to unseen long-horizon tasks in similar environmental settings, facilitated by the incorporation of a zero-shot planner framework, e.g., Socratic Models [5]. This framework effectively breaks down user-provided input into individual action steps, each serving as a distinct language command for our single-step RL agent, as illustrated in Figure 7. Following the execution of each command, the task space is reset, allowing for the subsequent command to be executed.

Apart from unseen colors, unseen letters are also included to evaluate the generalization capabilities of unseen scenarios. Table III demonstrates that the short-horizon ExploRLLM adapts to these settings, surpassing earlier Socratic Models versions. With the help of VLMs, which provide the bounding boxes and positions, our approach reformulates an observation space that helps RL to focus on discerning and learning the physical attributes of objects, which is essential for precise picking and placing tasks. This strategy effectively minimizes the potential distraction caused by variations in colors and shapes.

D. Zero-shot Transfer to the Real Robot

We conducted real-world evaluations of ExploRLLM under two scenarios: one replicating all letters in the simulation and another introducing the letter 'C', previously absent, with each scenario tested over 15 episodes. The short-horizon ExploRLLM achieved a success rate of 66.6% for seen letters and 53.3% for the scenario with an unseen letter. Meanwhile, the long-horizon ExploRLLM recorded success rates of 40% for seen letter scenarios and 33.3% for those including an unseen letter. Despite the Sim2Real gap, our approach demonstrates promising outcomes without any additional real-world training. As the VLM has already extracted the observation space, the RL agent trained within the simulation environment encounters fewer distractions from real-world noise. Figure 8 demonstrates the adaptability of our approach in managing diverse object orientations, understanding logical relationships between objects, and executing long-horizon tasks within real-world settings. However, the strategy continues to face challenges with noise in the color and depth perceptions of objects in real-world scenarios, which hinders the ability of the RL agent to manipulate objects. Employing a photorealistic simulator combined with thorough domain randomization is expected to enhance performance significantly.

TABLE III: Success rate (%) of short-horizon ExploRLLM with Socratic Models

Task Settings	Seen	Unseen Color	Unseen Letters
Socratic Models + ExploRLLM	74	68	56
Socratic Models + CLIPort	72	50	34

VIII. Conclusion and Discussion

In this work, we presented ExploRLLM, a method that combines RL with FMs. By utilizing actions informed by LLMs and VLMs for guiding exploration, we effectively speed up the convergence of RL, demonstrating the benefits of a synergistic approach that combines the strengths of both RL and FMs. We evaluated our proposed method

by conducting experiments involving tabletop manipulation tasks. We demonstrated its superior success rate by comparing our approach with policies solely based on LLMs and VLMs. Additionally, we showcased that the ExploRLLM policy can generalize to unseen colors, letters, and tasks. Our ablation experiments included training scenarios with different proportions of LLM-guide exploration, highlighting their significant impact on accelerating convergence. Additionally, we explored the ability to transfer the learned policy from simulation to the real world without further training, employing real robot experiments to validate this capability.

At present, our framework is primarily concentrated on tabletop manipulation tasks. We aim to broaden the scope of our framework to encompass a wider array of robotic manipulation applications. Furthermore, while our system is capable of correcting errors in low-level robotic actions, it encounters limitations in mitigating certain high-level errors that are less common in simulations. In the future, we intend to explore methods for addressing and rectifying these high-level discrepancies.

We also plan to enhance our method and utilize the Interactive Imitation Learning (IIL) paradigm [32] and actively query the user when the agent's prediction uncertainty is high, similar to [33].

REFERENCES

[1] N. Di Palo, A. Byravan, L. Hasenclever, M. Wulfmeier, N. Heess, and M. Riedmiller, "Towards a unified agent with foundation models," *arXiv preprint arXiv:2307.09668*, 2023.

[2] OpenAI, "Gpt-4 technical report," 2023.

[3] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147.

[4] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," in *Conference on Robot Learning*. PMLR, 2023, pp. 287–318.

[5] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022.

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

[7] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer, "Grounding large language models in interactive environments with online reinforcement learning," *arXiv preprint arXiv:2302.02662*, 2023.

[8] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy, "LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks," *arXiv preprint arXiv:2402.01817*, 2024.

[9] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

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

[13] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "PaLM-E: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[14] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

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

[16] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[17] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "ProgPrompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.

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

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[20] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," *arXiv preprint arXiv:2104.13921*, 2021.

[21] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," *arXiv preprint arXiv:2303.00001*, 2023.

[22] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik *et al.*, "Language to rewards for robotic skill synthesis," *arXiv preprint arXiv:2306.08647*, 2023.

[23] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma, "Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics," *arXiv preprint arXiv:2309.06687*, 2023.

[24] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[25] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas, "Guiding pretraining in reinforcement learning with large language models," *arXiv preprint arXiv:2302.06692*, 2023.

[26] E. Triantafyllidis, F. Christianos, and Z. Li, "Intrinsic language-guided exploration for complex long-horizon robotic manipulation tasks," *arXiv preprint arXiv:2309.16347*, 2023.

[27] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.

[28] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[30] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 12 348–12 355, 2021.

[31] B. van der Heijden, J. Luijkx, L. Ferranti, J. Kober, and R. Babuska, "Eagerx: Engine agnostic graph environments for robotics," *GitHub repository*, 2022.

[32] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, J. Kober *et al.*, "Interactive imitation learning in robotics: A survey," *Foundations and Trends® in Robotics*, vol. 10, no. 1-2, pp. 1–197, 2022.

[33] J. Luijkx, Z. Ajanovic, L. Ferranti, and J. Kober, "PARTNR: Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning," in *NeurIPS 2022 Workshop on Robot Learning*. arXiv.