Language-Guided Semantic Affordance Exploration for Efficient Reinforcement Learning

Runyu Ma

TU Delft

Language-Guided Semantic Affordance Exploration for Efficient Reinforcement Learning

by

Runyu Ma

Student Name	Student Number
Runyu Ma	5694337

Supervisors: Prof. Jens Kober,
Jelle Luijkx, Msc.
Dr. Zlatan Ajanović
Faculty: Faculty of Mechanical Engineering, Delft

TU Delft

Contents

Language-Guided Semantic Affordance Exploration for Efficient Reinforcement Learning

Runyu Ma[1], Jelle Luijkx[1], Zlatan Ajanović[2], and Jens Kober[1]

***Abstract*— Reinforcement Learning (RL) shows great potential for robotic manipulation tasks, yet it suffers from low sample efficiency and needs extensive exploration of state-action spaces. Some recent methods leverage the commonsense knowledge and reasoning abilities of Large Language Models (LLMs) to guide RL exploration toward more meaningful states. However, LLMs may generate semantically correct but physically infeasible plans, leading to unreliable solutions. In this paper, we propose *Language-Guided exploration for Reinforcement Learning* (LGRL), a novel framework that utilizes LLMs' planning capability to directly guide RL exploration. This approach utilizes LLM planning at both the task and affordance levels, enhancing learning efficiency by directing RL agents toward semantically meaningful actions. Unlike previous methods that rely on the optimality of LLM-generated plans or rewards, LGRL corrects sub-optimality and explores multimodal affordance-level plans without human intervention. We evaluated LGRL on pick-and-place tasks within standard RL benchmark environments, demonstrating significant improvements in both sample efficiency and success rates.**

I. INTRODUCTION

RL [1] provides a powerful framework for learning decision-making and control policies for robotics [2] through interactions with the environment. However, its practical application is often limited by low sample efficiency during exploration phases. Training a stable policy necessitates a thorough exploration of the state-action space and the collection of sufficient reward signals. This process can be particularly extensive for randomly initialized neural network policies, especially in complex exploration tasks.

Previous exploration methods provide intrinsic rewards to encourage exploration of novel states [3]–[6], which do not always align with meaningful robot behavior. To improve efficiency, earlier studies have integrated demonstrations that infuse human knowledge into off-policy RL training [7]–[11]. However, using demonstrations as a source of human knowledge can be costly, and the effectiveness of these methods heavily relies on the quality of the demonstrations.

In contrast, foundation models such as Llama3 [12] and GPT-4 [13], which are trained on large datasets, provide a robust alternative. These models, acting as approximate knowledge sources [14], utilize human-like reasoning to enhance robot manipulation tasks. Recent studies have shown that LLMs and Vision Language Models (VLMs) can comprehend environmental contexts and execute task-level reasoning, effectively translating high-level, open-language commands into sequences of actionable skills with human language patterns [15]–[19]. However, their reasoning capabilities can be limited by an incomplete understanding of the physical world, occasionally leading to errors.

Although foundation models may occasionally make errors, their approximation of human knowledge can significantly enhance RL training. Utilizing LLMs to generate dense reward functions [20]–[23] aligns robot actions with human language, effectively addressing challenges associated with sparse rewards. However, such human-like rewards can lead to unwanted behaviors, such as staying in regions with higher rewards rather than achieving task success. Beyond rewards, recent works [24], [25] explore using LLMs to generate direct actions, guiding robots towards semantically meaningful areas and altering the data distribution in off-policy RL replay buffers. These exploration-driven actions may start suboptimally but are refined during policy training. However, the effectiveness of these methods heavily depends on the quality of the actions produced by the LLMs and contrasts with off-policy RL approaches.

In this paper, we propose *Language-Guided exploration for Reinforcement Learning* (LGRL), a method that enables reinforcement learning agents to efficiently explore semantically meaningful regions guided by LLMs. At the affordance level—referring to the specific ways an object can be manipulated—actions can be multimodal, with some modes being semantically correct yet physically infeasible. Our method uses the values of the critics function to guide exploration at this level, helping the agent avoid infeasible actions at the affordance level.

We claim our contributions as follows:

1) We introduce LGRL, a method that uses LLMs to generate task- and affordance-level plans, guiding efficient exploration in reinforcement learning. This enhances the efficiency of reinforcement learning in manipulation tasks with sparse rewards by enabling robots to explore semantically meaningful action distributions.
2) We propose a method to correct errors and explore multi-modalities in affordance-level plans generated by foundational models. This approach guides the policy to explore semantic affordance spaces, effectively identifying the viable affordance modes.

II. RELATED WORK

In this section, we review methodologies in reinforcement learning and the application of foundation models in robotic manipulation.

[1] Cognitive Robotics, Delft University of Technology, The Netherlands (e-mail: {j.d.luijkx, j.kober}@tudelft.nl, r.ma-8@student.tudelft.nl).
[2] RWTH Aachen University, Germany (e-mail: zlatan.ajanovic@ml.rwth-aachen.de).

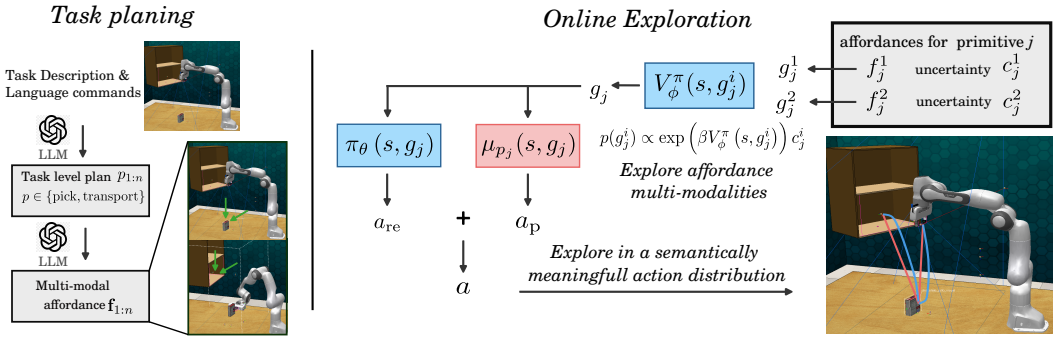


Fig. 1: **Language-Guided exploration for Reinforcement Learning** (LGRL). IGRL first leverages LLMs to generate task and affordance-level plans, directing RL to explore semantically meaningful regions with a goal state g and a base policy $\mu_{p_j}(s, g)$. Additionally, RL explores affordance-level multi-modalities using the goal-conditioned value function.

A. RL Exploration Methods

Classic exploration methods [3]–[6] in reinforcement learning reward agents with intrinsic rewards that encourage exploration of novel or uncertain states. These strategies aim to prevent agents from prematurely converging on exploitative actions, but in robotic manipulation, this exploration may accumulate non-contributory experiences. Demonstrations, representing human knowledge, are crucial in sparse reward environments and are incorporated into replay buffers to guide behavior through behavior cloning (BC) loss [7], [8], [10]. RLPD [9] utilizes high update-to-data (UTD) ratios and ensemble critics to guide training effectively using offline data without BC loss. IBRL [11] starts with demonstrations to train a policy, which then bootstraps the learning of a TD3 agent [26]. However, these methods still require high-quality and sufficient demonstrations to be effective.

B. Foundation Models in Manipulation

Researchers have demonstrated the ability of LLMs to do task-level reasoning for robotics. For instance, SayCan [16] and Grounded Decoding [27] integrate language scores with affordance scores from pretrained skills to decompose tasks. However, these methods may encounter errors as they do not explicitly consider spatial and logical reasoning. Several studies [28]–[30] have employed LLMs to adapt the Task and Motion Planning (TAMP) framework for task settings guided by open language commands. Furthermore, Plan-Seq-Learn [31] introduces a framework using LLMs for task-level planning, with a geometric planning module leading to initial conditions for the reinforcement learning process. These methods are effective in managing both contact-free and contact-rich tasks in robot manipulation.

C. Foundation Models and RL

LLMs have been effectively used to generate dense reward functions that align agent behavior with human language distribution [20]–[22]. While these rewards often enhance the relevance of agent actions, they can also lead to unintended behaviors, such as repeating actions for higher rewards without achieving the task or getting stuck in the early stages if those offer higher rewards than later ones. Eureka [23] proposes a method utilizing evolutionary optimization to refine reward code functions within the parallel environment but is less efficient due to the evolutionary process. Recent studies [24], [25] have explored generating direct actions to guide robots to semantically meaningful areas, altering data distributions in off-policy RL systems' replay buffers, with the need for high-quality actions to ensure effective guidance.

In contrast, our method, LGRL, learns a residual RL policy focused on semantically meaningful regions guided by LLMs. The residual action space allows the agent to optimize directly under LLM guidance with a more human-meaningful online data distribution. It leverages environmental interactions to address errors and explore multi-modalities at the affordance level of LLM guidance.

III. Preliminaries

A. Reinforcement Learning

In reinforcement learning, the problem is typically formulated as a Markov decision process (MDP), i.e., $\mathcal{M} \triangleq (S, A, R, P, \rho_0, \gamma)$, where S and A denote the state and action spaces respectively. The reward function R produces $r_t = R(s_t, a_t, s_{t+1})$, and the transition probability P defines $P(s_{t+1}|s_t, a_t)$. The initial state distribution ρ_0 and the discount factor γ are also specified.

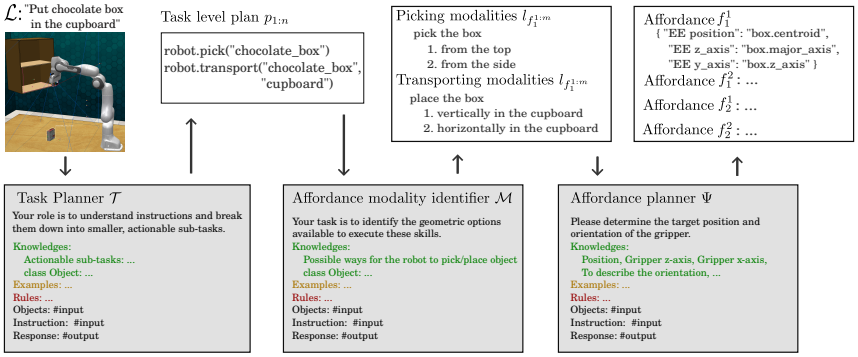
The goal of RL is to find an optimal policy π^* that maximizes expected cumulative rewards per episode, with π specifying action $a_t = \pi(s_t)$ for each state s_t:

$$\pi^* = \arg\max_\pi \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})\right]. \quad (1)$$

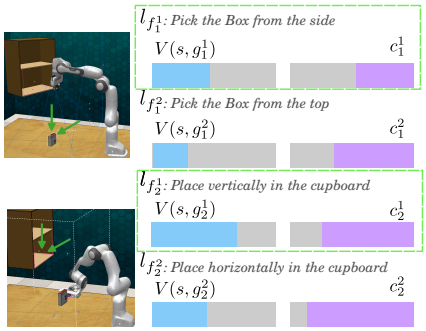
1) Value Function in RL: In reinforcement learning, value functions, often represented by neural networks with parameters ϕ, are trained to estimate the expected accumulated reward from a given state.

In on-policy RL methods, such as PPO [32], the value function $V_\phi^\pi(s)$ is used to estimate the expected reward when following a specific policy π from state s:

$$V_\phi^\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_t \gamma^t R(s_t, a_t, s_{t+1}) \middle| s_0 = s\right]. \quad (2)$$



(a) The LLM planning framework of LGRL comprises a task-level planner $\mathcal{T}$, an affordance modality identifier $\mathcal{M}$, and an affordance planner Ψ.



(b) LGRL explores affordance modalities with value $V(s, g)$ and uncertainty scores c.

Fig. 2: Visualization of the planning and exploration components in LGRL.

Off-policy RL methods, such as TD3 [26], utilize Q-functions $Q_{\phi_i}(s, a)$ to estimate the expected rewards for specific state-action pairs (s, a):

$$Q^{\pi}_{\phi_i}(s,a) = \mathbb{E}_{a\sim\pi(a|s)}\left[\sum_t \gamma^t R(s_t, a_t, s_{t+1}) \Big| s_0 = s,\ a_0 = a\right]. \tag{3}$$

In our approach, we leverage these value functions within the RL framework to guide exploration at the affordance level.

B. Problem Formulation

In our task settings, we concentrate on pick-and-place manipulation tasks similar to those described in [33], [34]. Our preliminary problem formulation defines the observation and action spaces as follows: The observation space comprises the robot's state and the states of objects in the environment. The action space A includes the end effector's relative pose and the gripper's open-close actions. The environment provides sparse external rewards $R_e : S \times A \to \{0, 1\}$, which indicates task success.

Our goal is to leverage the reasoning capacity of LLMs to generate plans that guide reinforcement learning exploration.

IV. Methods

This section presents the LGRL method, which utilizes the reasoning abilities of LLMs to direct RL agents toward semantically meaningful behaviors.

1) The process of guiding RL with LLM-generated plans is detailed in Section IV-A.
2) Foundation models convert high-level task instructions into a sequence of executable plans in task and affordance level, as described in Section IV-B.
3) The RL agent explores multi-modalities at the affordance level illustrated in Sec. IV-C.

A. RL with LLM Guidance

In this section, we discuss how plans generated by LLMs guide RL exploration efficiently. Before training, the planning pipeline generates plans at both the task and affordance levels. Task-level planning decomposes language commands into a sequence of n pick-transport primitives $p_{1:n}$, while affordance plans $\boldsymbol{f}_{1:n}$ describe the manipulation approach. The complete plan is represented as $\boldsymbol{p} = \{p_i(\boldsymbol{f}_j) \,|\, j = 1, \ldots, n\}$, where $p \in \{\text{pick}, \text{transport}\}$, as illustrated in Figure 2a.

During training, primitive p_j starts by translating the affordance-level plan into a goal state g_j, representing the SE(3) pose of the end-effector. A Proportional-Derivative (PD) controller serves as the base policy $\mu_{p_j}(s, g_j)$ to move toward g_j with linear trajectories in both position and orientation spaces. Actions a_p generated by these controllers aim to meet these goals. The RL agent learns a residual action policy $a_{\text{re}} = \pi_\theta(s, g)$. The residual policy allows LLMs to direct exploration towards semantically meaningful regions online while simultaneously training the RL to correct sub-optimalities and inaccuracies in LLM-controllers, as depicted in Figure 2a. Action a executed by the system is the sum of a_{p} and a_{re}:

$$a = a_{\text{p}} + a_{\text{re}}. \tag{4}$$

To guide exploration toward the goal state, the agent is rewarded with an intrinsic reward r_i that quantifies the distance to the goal state. This dense reward is general, depending only on the primitive used and not requiring fine-

Algorithm 1 *TaskPlan*

Input: Task description L
Output: Plans in task and affordance level $\{p_{1:n}\}$, $\{\boldsymbol{f}_{1:n}\}$
Component: LLM planner $\mathcal{T}$, $\mathcal{M}$, Ψ

$\{p_{1:n}\} \leftarrow \mathcal{T}(L)$ ▷ Decompose using planner $\mathcal{T}$
for $j \leftarrow 1$ **to** n **do**
  $\{l_{f_j^{1:m}}\} \leftarrow \mathcal{M}(p_j)$ ▷ Query LLM possible affordances modalities
  for $i \leftarrow 1$ **to** m **do**
    $f_j^i \leftarrow \Psi(p_j, l_{f_j^i})$ ▷ Plan each modalities
  end for
  $\boldsymbol{f}_j = \{f_j^1, \ldots, f_j^m\}$
end for
return $\{p_{1:n}\}$, $\{\boldsymbol{f}_{1:n}\}$

tuning for specific tasks:

$$r_i = R_i\left(e_{\text{pos}}, e_{\text{quat}}\right), \tag{5}$$

where e_{pos} is the positional error, and e_{quat} is the rotational error. Here, R_i represents a general dense reward function determined by errors relative to the goal state.

Our LLM-guided approach initializes the policy with a more semantically meaningful state distribution. Consequently, the RL agent needs only to fine-tune an optimal policy around the state distribution induced by the LLM-generated policy, greatly enhancing training efficiency. Moreover, our method can be applied to both on-policy and off-policy methods, as it does not depend on off-policy data in the replay buffer.

B. LLM Planning as Guidance

This section describes how LLMs convert high-level task instructions into task- and affordance-level plans. This pipeline consists of three LLM planners: a task-level planner $\mathcal{T}$, an affordance modality identifier $\mathcal{M}$, and an affordance-level planner Ψ.

The task-level planner $\mathcal{T}$ translates high-level task descriptions L into basic primitive actions, such as "Pick up the cube" or "Place the box in the cupboard". These actions are converted into Python primitives p, like 'robot.pick(object)', for reuse across multiple episodes.

Affordance-level plans are inherently multimodal in their semantics; however, while they may be semantically sound, not all modes are necessarily physically feasible. For example, the actions of picking and placing a box involve semantic ambiguities: a box could be grasped from the top or side and placed either horizontally or vertically. Placing a food box horizontally on a table may cause it to spill its contents, whereas placing it vertically might lead to an inaccessible configuration space. These challenges complicate the use of reinforcement learning for affordance-level guidance.

To address these challenges, we developed an affordance modality identifier $\mathcal{M}$ that queries the LLMs to generate all semantically feasible affordance plans when multimodal language characteristics are present. It generates descriptions $l_{f_j^{1:m}}$ for m different affordance modalities, such as "pick the box from the side" or "pick the box from the top".

The affordance-level planner $\Psi : (p_j, l_{f_j^i}) \rightarrow f_j^i$ then maps these descriptions to robot end-effector poses in natural language f_j^i, rather than precise SE(3) coordinates. For picking tasks, affordances are defined by three attributes: position, end-effector (EE) z-axis, and EE y-axis—specifying the direction the end-effector faces and the direction in which the gripper opens. This enables the LLM to align the robot's movements with environmental features effectively, as shown in Figure 2a. For transport tasks, object-centric formulations are used to accommodate variations between expected and actual picking poses, enhancing robustness and environmental adaptability.

C. Affordance Exploration

Our method is designed such that it explores affordance multi-modalities. In the training phase, affordance plans

Algorithm 2 *LGRL*

Input: Task description L
Output: Trained policy π_θ

$\{p_{1:n}\}, \{\boldsymbol{f}_{1:n}\} \leftarrow$ *TaskPlan*(L) ▷ Planning phase
Initialize policy π_θ and critics V_ϕ, uncertainty score c_j^i
for step in training_steps **do**
  if p_j needs to be initialized **then**
    for all f_j^i in $\boldsymbol{f}_j$ **do**
      $g_j^i \leftarrow$ *Parser*(f_j^i) ▷ Transfer to goal state
      $p(g_j^i) \propto \exp\left(\beta V_\phi^\pi\left(s, g_j^i\right)\right) c_j^i$ ▷ Calculate goal probabilities
    end for
    $g_j^i \sim p(g_j^i)$
    $c_j^i \leftarrow \max(c_j^i - \alpha c_j^i, c_{min})$ ▷ Update uncertainty score
  end if
  $a_{\text{re}} \leftarrow \pi_\theta(s, g_j^i)$
  Train(π_θ, V_ϕ) ▷ Train policy and critics
end for
return π_θ

$f_j^{1:m}$ are converted into goal states, $g_j^{1:m}$. The selection of a goal state g_j^i during training is determined by its alignment with the value function and an uncertainty score. The value function in RL provides an estimate of the expected return from the current state, making it an effective tool for approximating the value of a goal state based on current observations. The selection probability for on-policy reinforcement learning agents, where the value function is denoted as V_ϕ^π, is modeled as:

$$p(g_j^i) \propto \exp\left(\beta V_\phi^\pi\left(s, g_j^i\right)\right) c_j^i \tag{6}$$

and for off-policy agents using a Q-function $Q_\phi(s, a)$, it is given by:

$$p(g_j^i) \propto \exp\left(\beta Q_{\phi_1}\left(s, g_j^i, \pi\left(s, g_j^i\right)\right)\right) c_j^i \tag{7}$$

where $\beta > 0$ is a temperature parameter that controls the decision-making sharpness, and c_j^i is an uncertainty score that balances the exploration-exploitation trade-off. The uncertainty score is updated as follows:

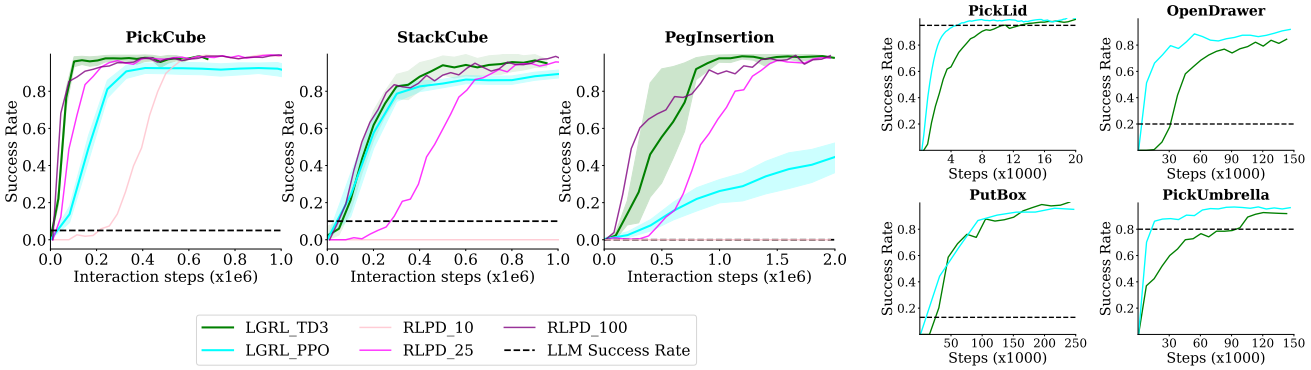
$$c_j^i \leftarrow \max(c_j^i - \alpha c_j^i, c_{\min}) \tag{8}$$

Here, α is a small positive constant significantly less than 1, and c_{min} serves as the lower bound for the uncertainty score.

In this way, we enable reinforcement learning agents to explore a higher semantic hierarchy, effectively addressing the issues associated with multi-modality at the affordance level. The complete algorithm is detailed in Algorithm 2.

V. Experiments In Simulation

We conduct experiments within simulation environments to demonstrate the effectiveness of our proposed framework.



(a) Performance of LGRL using TD3 and PPO in Maniskill tasks, evaluated using three different random seeds. A comparison is made between LGRL and the RLPD [9] method.

(b) Performance of LGRL using TD3 and PPO in RLbench tasks.

Fig. 3: Training curves in simulation environments, with dashed lines indicating the success rate of LLM policies.

A. *Environmental Setups and Results*

Our evaluation involves a diverse range of tasks from simulated environments: four from RLbench [33] and three from Maniskill3 [34]. The robot end-effector control uses relative position or velocity instead of joint space control, simplifying the action space for language models. This space includes delta velocity, delta orientation, and gripper actions, represented as $\boldsymbol{a} = (\delta_x, \delta_y, \delta_z, \delta_{rx}, \delta_{ry}, \delta_{rz}, \text{grip})$.

We learn policies with a residual action space built on top of base actions defined by pre-existing primitives. These primitives are implemented using a PD controller that moves linearly toward the target. The proportional gain is set to 1 by default, with movements constrained by maximum velocity and orientation velocity limits. For the pick primitive, the termination condition is met when the object's distance from its initial position exceeds a specified threshold. For the transport primitive, termination occurs when the positional error to the goal is below a threshold, and the object is nearly static.

We also design a general dense reward function for different primitives. For pick actions, the intrinsic reward is defined as:

$$r_i = -\tanh(w_{\text{pos}} \cdot e_{\text{pos}} + w_{\text{quat}} \cdot e_{\text{quat}}). \tag{9}$$

For transport actions, the reward is computed as follows:

$$r_i = 2 - 2 \cdot \tanh(w_{\text{pos}} \cdot e_{\text{pos}} + w_{\text{quat}} \cdot e_{\text{quat}}). \tag{10}$$

Here, w_p and w_q are weights applied to the positional error and the rotational error, respectively. In addition to the intrinsic reward r_i, an external reward r_e (where $r_e \gg r_i$) is provided by the environment, granted upon successful task completion or other specific termination conditions.

We chose OpenAI GPT-4o as the LLM due to its robust performance, affordability, and quick response times. To implement LLM planners, these planners follow a uniform structure that includes a brief task description, followed by specific instructions and task-related knowledge. While LLMs generally yield reasonable results, a suitable answer in one context may not necessarily be applicable in another.

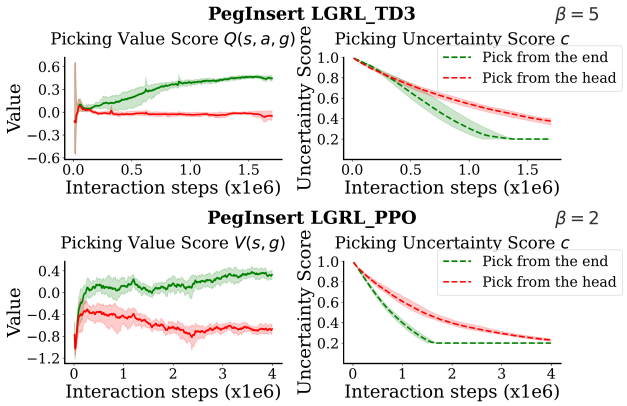


Fig. 4: PegInsertion Task: LGRL learns the correct picking position modality, i.e., picking from the end.

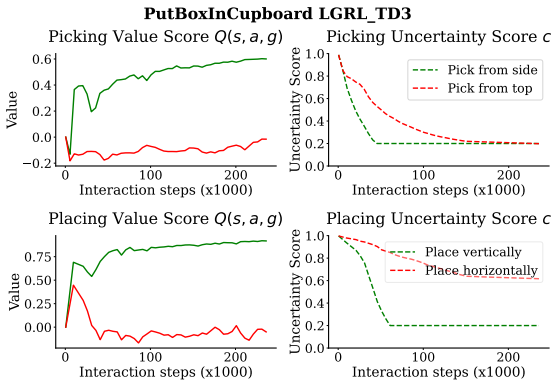
To address this, we provide several examples demonstrating the desired format and reasoning. Additionally, we use these planners to pre-query the LLM before the training phase, storing results as Python functions and dictionaries for reuse during training.

1) Maniskill Environments: In the Maniskill environments, we evaluate the LGRL framework using three representative pick-and-place tasks: PickCube, StackCube, and PegInsertionSide. This experiment aims to demonstrate the sample efficiency of our method by comparing it against three baselines: Text2Reward [22], Oracle reward, and RLPD [9].

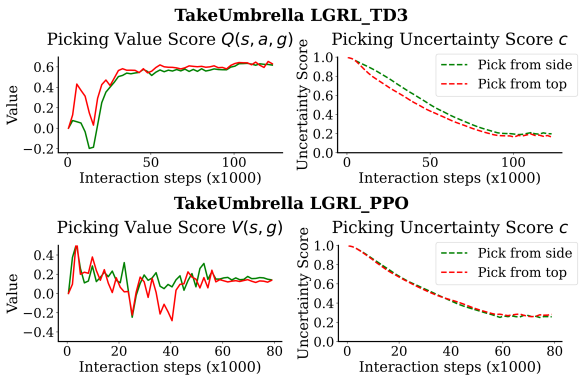
Text2Reward leverages the reasoning capabilities of LLMs to generate dense rewards that implicitly guide robots through necessary task steps, promoting meaningful behavior without human fine-tuning. Our implementation uses a zero-shot approach of Text2Reward without any prior expert reward examples.

Oracle reward utilizes the expert-designed reward function provided within the Maniskill environment.

RLPD is an effective reinforcement learning method that leverages demonstrations. It keeps a 50% offline training date



(a) PutBoxInCupboard task: identify pick-place modalities.



(b) TakeUmbrella task: learning both picking affordance modalities.

Fig. 5: Multimodal affordance exploration, showcasing value and uncertainty score curves across tasks.

data proportion and uses 10 critics to avoid overfitting. We tested it with 10, 25, and 100 demonstrations, training with sparse rewards.

While using expert rewards and dense rewards generated by LLMs demonstrates promising enhancements in episode rewards, no significant improvement in success rates is observed within a million steps, which is consistent with the results in [22]. Although Text2Reward successfully generates human-meaningful rewards, the lack of task-specific knowledge or RL expertise necessitates expert input in the prompt or human feedback to refine the reward functions. Furthermore, expert-designed rewards also failed to show improvement, indicating that methods relying solely on reward still require extensive environmental steps to learn a policy effectively.

The RLPD method exhibits improvements in success within a million steps and high sample efficiency with 100 demonstrations. However, their efficiency declines with fewer demonstrations and shows no improvement with only 10 for some tasks, as depicted in Figure 3a. Our experiments confirm that RL-with-demonstration methods heavily depend on demonstrations, while our method achieves comparable efficiency to RLPD with 100 demonstrations but without requiring human effort.

Our approach, which directly guides robots to goals generated by LLMs, achieves high convergence speeds and stable convergence across all tasks built on PPO and TD3, as shown in Figure 3a. The only exception is the PPO version of our method, which does not converge within 2 million steps but shows a positive trend. Although on-policy methods typically have lower sample efficiency than off-policy methods due to their inherent design, this performance gap is less noticeable in our approach. This efficiency is primarily due to the predefined primitives towards goals generated by LLMs that provide effective actions, aiding significantly in the policy optimization process. Consequently, the guidance from LLMs allows the policy optimization method to converge significantly faster than it would when optimizing a fully stochastic policy.

Table I shows the results of the affordance modality identifier for each task. The PickCube and StackCube tasks do not exhibit multiple options at the affordance level. For the PegInsertion task, the LLM-generated affordance plan is multi-modal, suggesting that the peg can be picked up from either the head or the end. Although achieving stable performance in PegInsertion typically requires millions of steps, our method successfully identifies the correct mode, as illustrated in Figure 4. Despite focusing on learning one mode, the method continues to explore alternative modes, as indicated by the uncertainty scores.

TABLE I: Summary of identified affordance modality across various tasks.

Tasks	N Modalities	Affordance Description
PickCube	1	–
StackCube	1	–
PegInsert	2	Pick: the peg from the end / head.
OpenDrawer	1	–
TakeUmbrella	2	Pick: the handle from top / side.
PutBoxIn Cupboard	4	Pick: the box from the side / top. Transport: vertically / horizontally in the cupboard.
TakeLid	1	–

2) RLbench Environments: RLbench provides a wide range of tasks that mimic everyday activities, making it a suitable platform for evaluating our LLM task planning and affordance-level planning pipeline. We assessed the LGRL method across four environments: PutBoxinCupboard, OpenDrawer, TakeUmbrellaOutOfUmbrellaStand, and TakeLidOffSaucepan. The control mode selected was EndEffectorPoseViaIK, which utilizes relative position and orientation for action.

Among these, we chose the PutBoxinCupboard task as our motivation example, as shown in Figure 2. This task is a modified version of the PutGroceriesInCupboard task, where the robot is required to place a box inside a cupboard. Additionally, we imposed a manual constraint to prevent the robot

from tilting the box, ensuring the contents do not spill. This constraint is not explicitly provided during the LLM planning phase but is incorporated as an external reward during RL training. Additionally, to ensure practical applicability, we integrated collision detection as another form of external reward signal. Our method aims to effectively explore both the affordance level and the execution level to learn a policy that successfully places the box in the cupboard.

During the planning phase, the LLM identifies multi-modalities within the task-level plan. The picking action offers two options: "pick from the side" and "pick from the top", while the placing phase also presents two possibilities: placing the object vertically or horizontally in the cupboard. As depicted in Figure 5a, as the value function learns the accumulated discounted rewards, the affordance-level plans learn to find the option with a higher success rate based on its critic's values. Through exploration, the system autonomously discovers that picking from the top may lead to an unreachable goal space for the transportation phase, and placing the box horizontally could violate the constraint on object orientation.

We conducted experiments on the four aforementioned tasks, with the training curve shown in Figure 3b. The results demonstrate that our method achieves high sample efficiency in both TD3 and PPO as the base policies. Notably, apart from the PutBoxinCupboard task, the TakeUmbrellaOutOfUmbrellaStand task also exhibits multimodalities in the task-planning phase. The LLM identifies these as "pick the umbrella from the top" and "pick the umbrella from the side". It is important to note that both plans are feasible, and RL successfully learns to adapt to these modalities as depicted in Figure 5b.

VI. Real World Experiments

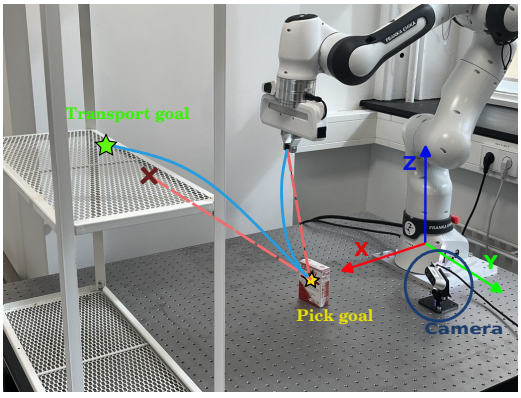
To demonstrate the effectiveness of our method in real-world applications, we evaluated it on a manipulation task, "PutBoxInCupboard" which requires stable object picking and collision avoidance during placement—an ideal scenario to showcase residual learning.

A. Experimental Setup

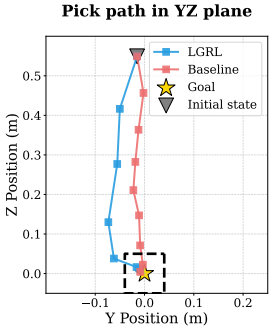
We utilized a Franka Emika Panda robot with a Franka gripper, controlled by a Cartesian impedance controller controlling the end-effector's position and orientation at 1kHz. Our method delivered end-effector pose commands at a lower frequency to the impedance controller. Real-time object tracking was facilitated using a RealSense D435 RGB-D camera, as depicted in Figure 6a.

B. Result

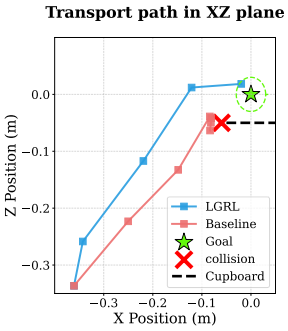
We deployed our algorithm with TD3 on the robot and conducted 15 episodes using our method and a baseline LLM controller that directly employed the plans generated by LLMs. Our method achieved a success rate of 93.3% with only one failure, while the LLM-only controller had a 0% success rate, failing all episodes due to collisions with the cupboard.



(a) Real world settings for task: PutBoxinCupboard



(b) Pick trajectory for PutBoxinCupboard



(c) Place trajectory for PutBoxinCupboard

Fig. 6: Visualization for real robot experiment

Figures 6b and 6c illustrate the detailed trajectories used to accomplish this task. As shown in these figures, the actions generated by the primitive policy are straightforward and directed toward the goal position defined by the LLM planner. Even without specific Sim-to-Real techniques, the base policy from these primitives guides the robot to regions with higher confidence, even when encountering unknown areas.

The residual policy refined trajectories for physical feasibility. In Figure 6b, the RL agent learns to avoid collisions during the picking phase. Similarly, in Figure 6c, it adjusts upward during placement to prevent collisions with the cupboard. Additionally, the robot moves in larger, stable steps, completing tasks more efficiently than LLM controllers. Overall, the residual policy significantly outperforms the LLM-only policy in both success rate and reliability.

VII. Conclusion And Discussion

In this paper, we present the LGRL method, which leverages the reasoning capabilities of LLMs to guide efficient exploration in reinforcement learning at both the execution and affordance levels. We evaluated our approach against baselines involving LLM-guided reward shaping and efficient RL with demonstration in simulated environments. The results demonstrate that our method enables LLMs to generate plans and detect multi-modalities at the affordance level, thereby significantly enhancing the efficiency of rein-

forcement learning and outperforming baselines in terms of success rate and sample efficiency.

However, our approach has limitations. The current planning framework cannot handle complex objects with intricate geometries and is limited to simpler objects such as cubes, boxes, or drawer handles. Additionally, our method relies on access to object states, which poses challenges for real-world deployment.

To address these limitations, we plan to incorporate interactive learning, where humans provide one or a few demonstrations to help the robot plan for objects with more complex geometric relationships. To improve generalizability, we also aim to use manipulation foundation models, such as AnyGrasp [35], to generate affordance goals without relying on precise state information of the objects.

REFERENCES

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

[2] 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.

[3] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015.

[4] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, 2016.

[5] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," *arXiv preprint arXiv:1703.01732*, 2017.

[6] G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos, "Count-based exploration with neural density models," in *International conference on machine learning*. PMLR, 2017, pp. 2721–2730.

[7] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[8] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[9] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1577–1594.

[10] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[11] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation bootstrapped reinforcement learning," *arXiv preprint arXiv:2311.02198*, 2023.

[12] W. Huang, X. Ma, H. Qin, X. Zheng, C. Lv, H. Chen, J. Luo, X. Qi, X. Liu, and M. Magno, "How good are low-bit quantized llama3 models? an empirical study," *arXiv preprint arXiv:2404.14047*, 2024.

[13] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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," in *International Conference on Machine Learning*. PMLR, 2023, pp. 8657–8677.

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

[22] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Automated dense reward function generation for reinforcement learning," *arXiv preprint arXiv:2309.11489*, 2023.

[23] 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.

[24] R. Ma, J. Luijkx, Z. Ajanovic, and J. Kober, "Explorllm: Guiding exploration in reinforcement learning with large language models," *arXiv preprint arXiv:2403.09583*, 2024.

[25] L. Chen, Y. Lei, S. Jin, Y. Zhang, and L. Zhang, "Rlingua: Improving reinforcement learning sample efficiency in robotic manipulations with large language models," *IEEE Robotics and Automation Letters*, 2024.

[26] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[27] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman *et al.*, "Grounded decoding: Guiding text generation with grounded models for robot control," *arXiv preprint arXiv:2303.00855*, 2023.

[28] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

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

[30] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[31] M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov, "Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks," *arXiv preprint arXiv:2405.01534*, 2024.

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

[33] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[34] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, "Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations," *arXiv preprint arXiv:2107.14483*, 2021.

[35] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics*, 2023.

A

Acknowledgement

First and foremost, I extend my deepest gratitude to my thesis supervisors, Jelle, Zlatan, and Jens. I feel incredibly fortunate to have been guided by such knowledgeable and supportive mentors. Your expert advice was instrumental throughout my thesis project and in introducing me to the world of science. Moreover, your patience and support were invaluable. I have always been less confident in my English communication skills, so I am particularly thankful for your patience during our discussions and your assistance with these issues.

I am also immensely grateful to my parents for their support over the past 2 years and 22 years before that. Thank you for your love, care, and guidance.

My heartfelt thanks go to my friends in China and Delft, whose companionship and love have been tremendously meaningful and motivating.

Especially, I would like to express my gratitude to my girlfriend, Yufei, for her companionship, love, and understanding.

This journey would not have been possible without all of your support. Thank you.

ExploRLLM: Guiding Exploration in Reinforcement Learning with Large Language Models

Runyu Ma[*1], Jelle Luijkx[*1], Zlatan Ajanović[2], and Jens Kober[1]

***Abstract*— In robot manipulation tasks with large observation and action spaces, reinforcement learning (RL) often suffers from low sample efficiency and uncertain convergence. As an alternative, foundation models have shown promise in zero-shot and few-shot applications. However, these models can be unreliable due to their limited reasoning and challenges in understanding physical and spatial contexts. This paper introduces ExploRLLM, a method that combines the commonsense reasoning of foundation models with the experiential learning capabilities of RL. We leverage the strengths of both paradigms by using foundation models to obtain a base policy, an efficient representation, and an exploration policy. A residual RL agent learns when and how to deviate from the base policy while its exploration is guided by the exploration policy. In table-top manipulation experiments, we demonstrate that ExploRLLM outperforms both baseline foundation model policies and baseline RL policies. Additionally, we show that this policy can be transferred to the real world without further training. Supplementary material is 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], can generate commonsense-aware reasoning in various scenarios. For instance, LLMs have demonstrated zero-shot planning capabilities [3], breaking down complex tasks into detailed step-by-step plans without additional training. When integrated with Vision-Language Models (VLMs), LLMs leverage cross-domain knowledge for robot perception and planning in manipulation tasks [4]. This synergy allows for the extraction of environmental affordances and constraints, forming a foundation for subsequent robotic planning [5]. Despite the impressive results of FMs, unpredictable failures in LLM predictions can still lead to robotic errors, and LLMs generally do not learn from past experiences [6], [7].

Reinforcement Learning (RL) offers a powerful framework for learning decision-making and control policies through interaction with the environment [8]. However, RL struggles with the "curse of dimensionality," where large observation and action spaces slow exploration and convergence. To address this, we propose combining FMs and RL, using FMs to guide the RL agent's exploration. While actions generated by FMs may be sub-optimal or fail, they highlight meaningful regions in the action space for exploration. Traditional RL exploration strategies (e.g., ϵ-greedy, Boltzmann exploration [9]) are stochastic, focusing on exploration-exploitation trade-offs, but lack mechanisms to incorporate prior knowledge for faster convergence. We instead use LLMs as few-shot planners, generating actions that serve as exploration steps in RL, increasing the likelihood of successful states and gathering more relevant state-action pairs for off-policy RL agents.

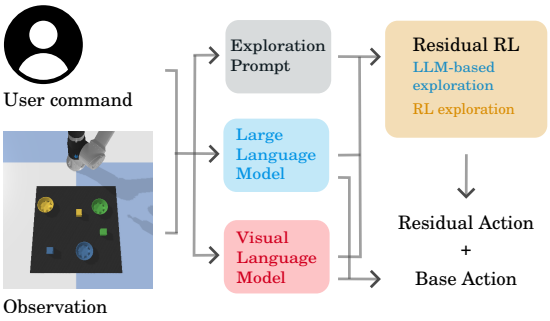


Fig. 1: Graphical overview of ExplorLLM.

Our method, ExploRLLM, improves performance by compensating for FMs' sub-optimality and biases through RL, while FMs accelerate RL training by reducing observation spaces and guiding exploration. To summarize, our main contributions are: 1) We propose ExploRLLM, which employs an RL agent with a) residual action and observation spaces based on affordances identified by FMs and b) LLM-guided exploration. 2) We introduce a prompting method for LLM-based exploration using hierarchical language-model programs, leading to faster convergence. 3) We show that ExploRLLM outperforms policies derived solely from LLMs and VLMs and generalizes to unseen scenarios, tasks, and real-world settings without additional training.

II. RELATED WORK

A. Foundation Models for Planning in Robotics

Researchers have shown that LLMs can generate zero-shot or few-shot plans using reasoning capabilities [3], [10], which is crucial for high-level planning in robotics. These models facilitate task-level planning by integrating environmental groundings, such as affordance value scores [11] or feedback [12], with their language groundings. Furthermore, LLMs can generate robot-centric code programs as representations for both task-level [13] and skill-level planning [14].

[*] Equal Contribution. [1] Cognitive Robotics, Delft University of Technology, The Netherlands (e-mail: {j.d.luijkx, j.kober}@tudelft.nl, r.ma-8@student.tudelft.nl). [2] RWTH Aachen University, Germany (e-mail: zlatan.ajanovic@ml.rwth-aachen.de).

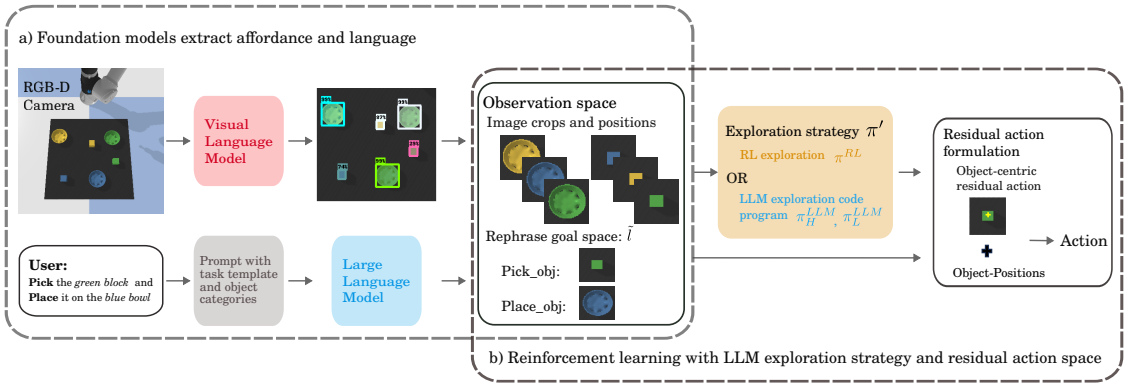


Fig. 2: Implementation structure of ExplorLLM for tabletop manipulation, combining the strengths of RL and FMs.

VLMs are increasingly integrated into robotics as a perception module of environmental context. The integration of knowledge from LLMs and VLMs facilitates the creation of perception-planning pipelines [4] and the construction of 3D value maps for zero-shot planning frameworks [5]. However, directly applying VLMs and LLMs to zero-shot tasks may not guarantee success or safety due to real-world uncertainty. In our research, we treat these actions as exploratory behaviors within an RL framework.

B. Foundation Models and Reinforcement Learning

Incorporating FMs into RL frameworks has notably improved RL's effectiveness. In [15], 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 [16], self-refinement [17] and evolutionary optimization over reward code to enable complex tasks such as dexterous manipulation [18]. Regarding exploration, authors in [19] 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 [20]. Contrary to these studies, our approach employs LLM-generated code policies to guide exploratory actions rather than focusing on reward shaping.

III. Problem Formulation

In this study, we focus on tabletop manipulation tasks to implement and evaluate our ExplorLLM method, as these tasks facilitate the creation of a base policy using FMs. We use a VLM for object detection and an LLM to identify the object to be manipulated from a command. Each manipulation task begins with a linguistically described goal, denoted by $\boldsymbol{l}$. At each time step t, the agent receives an observation $\boldsymbol{o}^t$, consisting of an overhead RGB-D image and the state of the end-effector. Similar to existing methods (e.g., Transporter [21]), the action space involves a pick and a place primitive, denoted as $\{\mathcal{P}_{\text{pick}}, \mathcal{P}_{\text{place}}\}$, with each action parameterized by pick and place positions in a top-down view. We simplify this to a single motion primitive—either pick or place. This simplification makes the RL challenge more tractable by eliminating the need to learn a feature representation for each primitive individually. The pick or place action is defined as a tuple containing the primitive index k (0 for pick, 1 for place) and a top-down view position, expressed as $\boldsymbol{x}$, i.e., $\boldsymbol{a}^t = (k^t, \boldsymbol{x}^t)$. At each time step, the agent receives a reward r consisting of a dense reward component r^d and a sparse reward r^s.

IV. Framework: ExplorLLM

A. Observation and Action Spaces

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 pick object in a template like *"Put [pick_object] on the [place_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 the image in raw observation space $\boldsymbol{o}^t$, represented by their locations $\boldsymbol{X}_{\text{vlm}} = [\boldsymbol{x}_{\text{vlm}_1}, \boldsymbol{x}_{\text{vlm}_2}, ...]$. RGB-D visual inputs are segmented into crops based on bounding box positions, denoted as $\boldsymbol{M}_{\text{vlm}} = [\boldsymbol{m}_{\text{vlm}_1}, \boldsymbol{m}_{\text{vlm}_2}, ...]$. This method improves the system's robustness to detection inaccuracies and varying object shapes. The interpreted commands $\tilde{\boldsymbol{l}}$, the positional data $\boldsymbol{X}_{\text{vlm}}$ and the image patches $\boldsymbol{M}_{\text{vlm}}$ are then integrated into the reformulated RL observation $\boldsymbol{s}^t$.

As the VLM already extracts each object's position $\boldsymbol{X}^t_{\text{vlm}}[i^t]$, 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 k, an object index i and a residual position $\boldsymbol{x}_{\text{res}}$, expressed as $\tilde{\boldsymbol{a}}^t = (k^t, i^t, \boldsymbol{x}^t_{\text{res}})$, where $\boldsymbol{x}^t = \boldsymbol{X}^t_{\text{vlm}}[i^t] + \boldsymbol{x}^t_{\text{res}}$. This residual action allows to pick or place objects at specific locations. This is, for example, needed when picking the letter O, and $\boldsymbol{X}^t_{\text{vlm}}[i^t]$ denotes the center of the bounding box. In this case, the

Algorithm 1 Exploration strategy π^{E}

Input: state $\boldsymbol{s}^t$, 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 $j \leq \epsilon$ **then**
    Run LLM-generated high level action policy π_H^{LLM}
$$\boldsymbol{a}_H^t = (k^t, i^t) = \pi_H^{\mathrm{LLM}}(\boldsymbol{s}^t)$$
    Run LLM-generated low level action policy π_L^{LLM}
$$\boldsymbol{x}_{\mathrm{res}}^t = \pi_L^{\mathrm{LLM}}(\boldsymbol{s}^t, \boldsymbol{a}_H^t)$$
$$\tilde{\boldsymbol{a}}^t = (k^t, i^t, \boldsymbol{x}_{\mathrm{res}}^t)$$
else
    Run the reinforcement learning policy π^{RL}
$$\tilde{\boldsymbol{a}}^t = \pi^{\mathrm{RL}}(\boldsymbol{s}^t)$$
end if
return action $\tilde{\boldsymbol{a}}^t$

residual action $\boldsymbol{x}_{\mathrm{res}}^t$ is needed to prevent picking the letter O at its empty center.

B. LLM-Based Exploration

Traditional deep RL algorithms (e.g., SAC [22], PPO [23]) do not inherently promote frequent visits to high-value states in high-dimensional state-action spaces, making vision-based tabletop manipulation tasks particularly challenging. In such cases, RL agents may struggle when successful outcomes are rare. Leveraging the planning capabilities of LLMs and the perception strengths of VLMs can help guide the exploration process more effectively by tapping into the rich prior knowledge within these FMs. The LLM-based exploration strategy, denoted as π^{E} 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.

Prior research frequently prompts LLMs at every step to create plans for robotic manipulation, making the process highly resource-intensive. This method incurs significant time and financial costs due to the numerous LLM invocations required to train a single RL agent. Drawing inspiration from CaP [14], our methodology employs the LLM to hierarchically generate language model programs, which are then executed iteratively during the training phase as exploratory actions. 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.

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 3.

V. Implementation

The main components for the implementation of ExploR-LLM 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 [24]. We employ two convolutional layers to transform every image patch into a vector $\boldsymbol{\phi} \in \mathbb{R}^{n \times d}$, where n is the number of objects captured by VLM and d 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 $\boldsymbol{\phi}' \in \mathbb{R}^{n \times d'}$, where d' denotes the dimension of each patch's vector following encoding and concatenation. It then goes to a self-attention layer. The output features from this layer then 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 [25], 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]. It 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 [14], where LLMs have been shown capable of generating novel policy codes with example codes and commands, our prompt also includes examples. They are designed to guide the LLM in formulating plans and conducting geometric reasoning for our specific task scenarios.

For low-level exploration actions, we employ GPT-4 with Vision [2], which generates code using prompts that combine

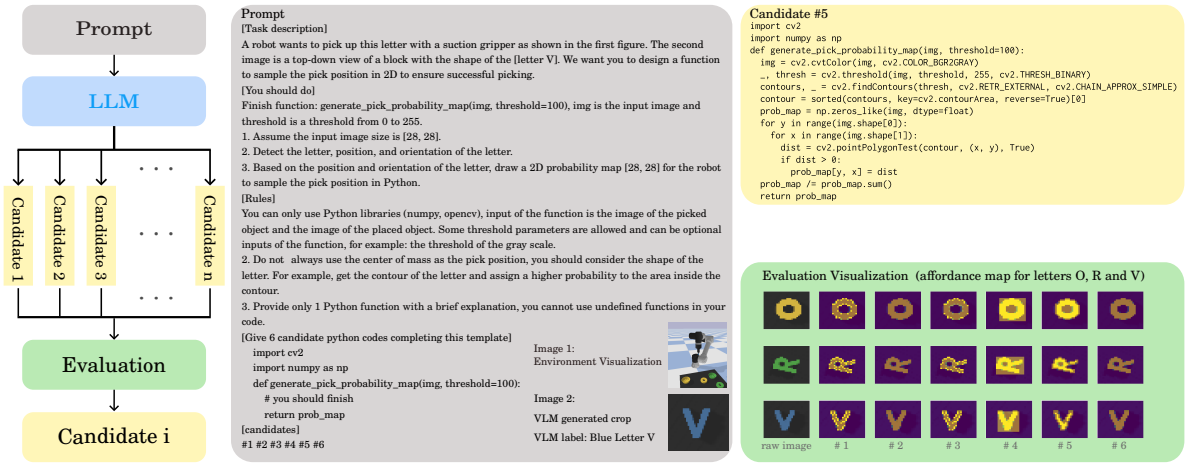


Fig. 3: Based on an exploration prompt, candidate policy code is generated. The exploration policy is selected after evaluation.

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 3, 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 sub-optimality, we use a stochastic policy based on the affordance map instead of a deterministic one that selects the point of highest affordance. Since RL improves through rewards from environmental interactions, sub-optimal exploration policies can be corrected via learning. This approach also allows for the generation of counter-examples during replay buffer collection.

VI. Experimental Setups

A. Simulation Setup

The proposed method is trained and evaluated on a simulated tabletop pick-and-place task, as shown in Figure 2. Similar to [21] and [26], we use a UR5e, and the input observation is a top-down RGB-D image. Inspired by [26], we increased the task difficulty by replacing simple blocks with various objects, such as letters. We assess our method in 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"*, as shown in Figure 6a. In the SH task, each episode starts with three letters and three bowls randomly placed on the table, with pick-and-place actions generated from random language commands. The task is completed when the robot accurately places the chosen letter in the specified bowl. In the LH task, all letters and bowls are randomly arranged, and the task is completed when each letter is placed in a bowl that matches its color.

B. Real-World Setup

We validated our approach on a Franka Panda robot equipped with a suction gripper and an RGB-D camera, as shown in Figure 6a, implementing our policy and code in the EAGERx [27] framework. 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. We used ViLD for bounding box identification based on object names. To simulate real-world conditions more accurately, we introduced noise to the bounding box center's position during the training phase in the simulation, mimicking the positional uncertainty inherent in VLM detection. We also added noise to bounding box positions and image inputs, simulating VLM detection uncertainty and camera noise, including lighting variations.

VII. Results

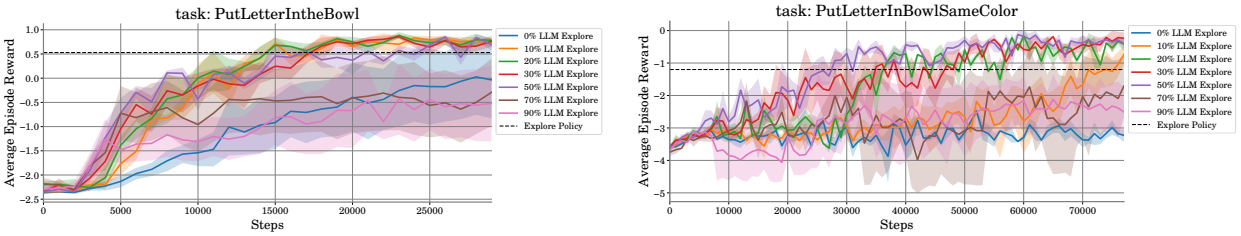
A. Simulation Results

We investigated the effect of varying LLM-based exploration frequencies on training convergence, using $\epsilon \in \{0.0, 0.1, \ldots, 0.9\}$, as shown in Figure 4. An ϵ of 0 corresponds to stardard SAC. Training was conducted with six random seeds per frequency, and each session began with a 20,000-step warm-up phase without LLM exploration, as no significant policy improvements were observed during this phase. Post-warm-up results, shown in Figure 4 and detailed in Table II for both short- and long-horizon tasks, indicate that ExploRLLM consistently outperforms LLM-only policies across various exploration frequencies.

In the short-horizon task (Figure 4a), training without LLM-based exploration is often unstable, resulting in either a successful policy or failure to converge. When the exploration frequency is within $0 < \epsilon \leq 0.5$, training stabilizes and converges more quickly, with minimal variation across different ϵ values. However, increasing ϵ beyond 0.5

TABLE I: Results of 50 evaluation episodes for short-horizon (SH), long-horizon (LH), and different initialization methods: no object overlap (NO) and allowed overlap (AO). ExploRLLM standard deviations are shown for 6 seeds.

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) *Pick the [pick letter] and place it in the [place color] bowl* (SH). (b) *Put all letters in the bowl of the corresponding color* (LH).

Fig. 4: Training curves for varying exploration rates in SH and LH tasks. ExploRLLM outperforms the exploration policies (dashed lines) and RL without LLM-based exploration ($\epsilon = 0$). In the LH task, LLM-based exploration is crucial for success.

TABLE II: ExploRLLM training returns for varying 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

TABLE III: Success rate (%) of SH ExploRLLM with [4].

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

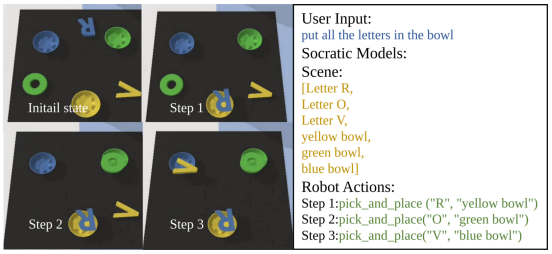


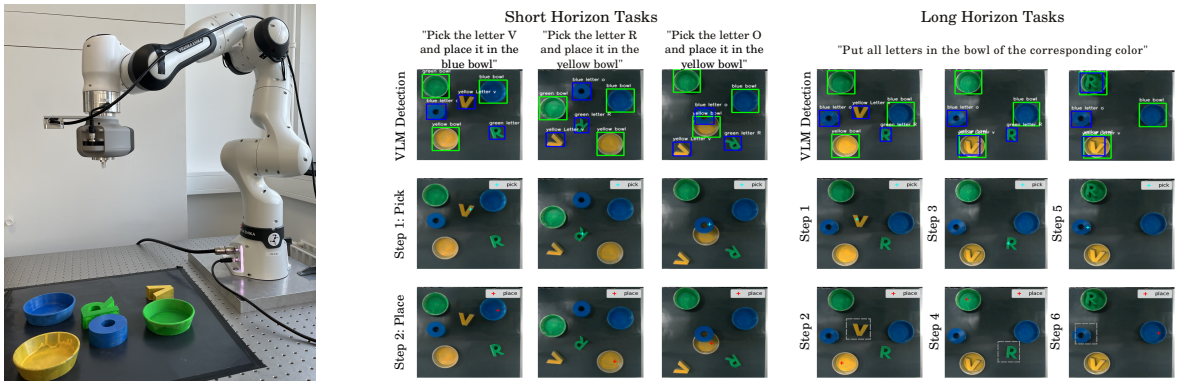
Fig. 5: Short-horizon ExploRLLM policies can be used in long-horizon tasks with zero-shot LLM planners, e.g. [4].

reduces the proportion of online data, slowing progress and introducing greater instability into the training. For long-horizon tasks, Figure 4b shows that higher frequencies of LLM-based exploration ($0 < \epsilon \leq 0.5$) correlate with faster training. These results highlight the importance of LLM-based exploration in navigating complex tasks by guiding experience toward the optimal region, mitigating challenges from large observation and action spaces. However, similar to the short-horizon tasks, excessive exploration rates introduce instability and slow convergence.

To evaluate the effectiveness of ExploRLLM, we benchmark its performance against four baselines: ExploRLLM without the LLM-based exploration policy, the CaP-style policy [14] (our exploration policy), Socratic Models [4], and Inner Monologue [12]. Our implementations of Socratic Models and Inner Monologue use ViLD [25] as the object detector and GPT-4 [2] as a multi-step planner. The individual steps are executed by a pre-trained CLIPort [26] model with 500 demonstrations. The key difference between Socratic Models and Inner Monologue is that Inner Monologue features a success detector that can identify mistakes.

During evaluation, the letter colors range from seen to unseen colors. Tasks and initialization methods vary, with "NO" indicating no overlap between the initial positions of letters and bowls, and "AO" allowing overlaps. These configurations assess each method's robustness in handling complex object relationships.

For short-horizon tasks, as shown in Table I, ExploRLLM maintains stable performance, whereas versions without the exploration policy often fail to converge and exhibit high variance in success rates and low-level errors. Our method surpasses LLM-generated policies in success rates, reduces robot behavior errors, and minimizes the performance gap between NO and AO scenarios, emphasizing the exploration policy's role in correcting FMs' inaccuracies. In contrast, CLIPort-based methods struggle with novel scenarios or complex geometric object relationships. For long-horizon tasks, RL agents without LLM-based exploitation fail to converge. As shown in Table I, ExploRLLM outperforms Socratic Models, Inner Monologue, and LLM-generated poli-



(a) Real-world experimental setup.

(b) Visualization of VLM detections and pick and place actions.

Fig. 6: ExploRLLM can be practically applied using a sim-to-real approach with transfer due to VLM object detections.

cies, achieving superior results in long-horizon tasks.

Although our short-horizon agent is trained specifically for a pre-defined pick-and-place task, our approach can transfer to unseen long-horizon tasks in similar environments. This is made possible by integrating a zero-shot planner framework, such as Socratic Models [4]. 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 5. 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. Using VLMs to provide bounding boxes and positions, our approach reformulates the observation space, enabling RL to focus on learning the physical attributes of objects, which is crucial for precise pick-and-place tasks. This strategy minimizes distractions from variations in colors and shapes.

B. Real-World Results

We conducted real-world evaluations of ExploRLLM in two scenarios: one replicating all letters from the simulation experiments and another introducing the previously unseen letter 'C', with each scenario tested over 15 episodes. The short-horizon ExploRLLM achieved success rates of 66.6% for seen letters and 53.3% for the unseen letter scenario. In comparison, the long-horizon ExploRLLM recorded success rates of 40% for seen letters and 33.3% for unseen letters. Despite the Sim2Real gap, our approach shows promising results without additional real-world training. As the VLM extracts the observation space, the RL agent trained in simulation is less distracted by real-world noise. Figure 6b illustrates the adaptability of our method in handling diverse object orientations, understanding logical relationships between objects, and executing long-horizon tasks in real-world settings. However, challenges remain with noise in the color and depth perception of objects, which hampers the RL agent's ability to manipulate objects. Using a photorealistic simulator with extensive domain randomization is expected to improve performance.

VIII. Conclusion and Discussion

In this work, we present ExploRLLM, a method that combines RL with FMs. By using actions informed by LLMs and VLMs to guide exploration, we accelerate RL convergence, demonstrating the benefits of integrating the strengths of both RL and FMs. We evaluated our method on tabletop manipulation tasks, showing superior success rates compared to policies based solely on LLMs and VLMs. ExploRLLM also generalizes to unseen colors, letters, and tasks. Ablation experiments with varying levels of LLM-guided exploration further highlighted its significant role in speeding up convergence. Additionally, we validated the method's ability to transfer learned policies from simulation to real-world scenarios without additional training through real robot experiments. Currently, our framework focuses on tabletop manipulation, but we plan to extend it to a broader range of robotic manipulation tasks. While the system can correct low-level robotic actions, it struggles with mitigating high-level errors that are less frequent in simulations. Future work will focus on addressing these high-level discrepancies.

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. 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] B. van der Heijden, J. Luijkx, L. Ferranti, J. Kober, and R. Babuska, "Engine agnostic graph environments for robotics (eagerx): A graph-based framework for sim2real robot learning," *IEEE Robotics & Automation Magazine*, pp. 2–15, 2024.

Guiding Exploration in Reinforcement Learning with Large Language Models

RO57010: Literature Research

Runyu Ma

Delft University of Technology

TU Delft

Guiding Exploration in Reinforcement Learning with Large Language Models

by

Runyu Ma

Student Name	Student Number
Runyu Ma	5694337

Supervisors: Jelle Luijkx, Dr. Zlatan Ajanovic, Dr. Jens Kober
Project Duration: April, 2024 - June, 2024
Affiliation: Dept. Cognitive Robotics, Faculty of Mechanical Engineering, TU Delft

18
TU Delft

Preface

This literature study report is the primary output of the RO57010 Literature Research course and forms an integral component of the author's master thesis work at the Department of Cognitive Robotics, Faculty of Mechanical Engineering, Delft University of Technology. The report is dedicated to studying reinforcement learning techniques for robot manipulation tasks, with a specific focus on utilizing foundation models to guide reinforcement learning exploration.

Disclaimer regarding LLM usage: I hereby affirm that I utilize ChatGPT solely for refining and polishing my existing literature study report. ChatGPT is not employed for generating new ideas or composing extensive paragraphs on my behalf. All ideas, concepts, and original content within my report are generated through my own research and understanding of the subject matter. ChatGPT serves only as a tool for linguistic refinement and does not contribute to the conceptualization or generation of content within my report.

Runyu Ma
Delft, September 2024

Contents

1 Introduction

1.1. Backgrounds and Social Impacts

Robot manipulation has great significance for helping people in industrial, daily life, and medical contexts. In industrial settings, robots handle tasks that are too hazardous or repetitive for humans, thereby boosting productivity and worker safety. Additionally, robots facilitate daily activities in homes and hospitals, taking over routine tasks to free up human time for more critical things.

In recent years, the application of machine learning techniques in manipulation has seen a significant rise. Technologies such as imitation learning and reinforcement learning empower robots to learn from experiences and adapt autonomously to new and intricate tasks. This adaptability proves invaluable in dynamic and unpredictable environments, like adaptive manufacturing or interactive services, where robots work alongside humans.

The advancement of robot learning techniques offers a broader perspective in these fields. With machine learning algorithms, robots are becoming increasingly capable of intelligent and adaptable coexistence with humans. This evolution enables robots to become integral, supportive companions in daily human activities, enhancing both the quality and efficiency of everyday life.

1.2. Research Focus

Reinforcement Learning (RL), as outlined in [91], offers a promising framework for decision-making through dynamic interactions with the environment. It is also utilized in developing control policies for robotics, as detailed in [48]. RL is particularly effective in addressing complex robot manipulation tasks involving contact-rich environments, such as pick-and-place operations and tool assembly, where dynamics are often unknown or difficult to model accurately [20]. Through continuous interaction with the environment, RL optimizes control policies based on objectives like success rate, accuracy, time efficiency, trajectory smoothing, or safety, all of which are encapsulated in the reward signal.

Despite its promising potential for training control policies in robotics, applying RL in robotic tasks, especially those requiring long-horizon manipulation, presents significant challenges due to the complexities of exploration in continuous action spaces. Robots frequently need to explore vast action spaces for millions of steps, often struggling to develop a stable policy that consistently reaches the optimal state distribution. To overcome these challenges, the development and implementation of effective exploration algorithms are crucial for successful manipulation tasks. Effective exploration algorithms are crucial for successful reinforcement learning, as they must broadly identify regions of high accumulated rewards and gather meaningful experience/data in these areas to optimize the policy. Balancing these two aspects of exploration is particularly challenging, especially for long-horizon manipulation tasks that require explicit learning of low-level control policies and implicit decision-making at the task level.

On the other hand, foundation models offer a promising approach for robot manipulation that eliminates the need to start learning from scratch by leveraging past experiences. Foundation models (FMs)[17], such as Large Language Models (LLMs) and Vision-Language Models, are trained on vast amounts of data and have shown considerable promise in the field of robotics. For instance, in some scenarios, LLMs like GPT-4[73] can generate human-like, commonsense-aware reasoning.

This ability has been effectively utilized as a zero-shot planner [39], capable of decomposing complex tasks into detailed, step-by-step plans without additional training. In robot manipulation, LLMs have demonstrated their potential to generate task-level plans [3, 38, 56, 37] or to provide knowledge for low-level actions [40, 103, 61], showcasing their versatility and effectiveness in enhancing robotic capabilities.

However, despite their potential, foundation models like LLMs often lack detailed knowledge of the physical world and specific tasks, which can lead to errors in certain scenarios. This limitation underscores that there is still a significant journey ahead before these models can be fully deployed in real-world robotics. For LLMs to be effectively used in practical applications, they require further development to enhance their understanding of the world and acquire more detailed knowledge about task environments. This improvement is crucial for handling the uncertainties inherent in real-world settings.

We are interested in exploring the integration of reasoning capabilities and embedded prior knowledge of FMs with the trial-and-error nature of RL. On the one hand, the prior knowledge contained within foundation models could significantly accelerate RL's exploration and training phases, eliminating the need for an exhaustive exploration of the action space and learning from scratch. On the other hand, RL, through its trial-and-error approach, could provide real-world grounding to LLMs. It also can correct mistakes and address sub-optimalities in the plans generated by foundation models through policy optimization. By integrating foundation models and RL, we can potentially mitigate the limitations of both approaches, enhancing their effectiveness and applicability in complex environments.

Based on the reasoning outlined above, the research question for this literature study is formulated as follows:

How can Foundation Models be effectively incorporated into deep reinforcement learning exploration for robot manipulation to enhance their training efficiency?

1. What are the state-of-the-art (SOTA) reinforcement learning methods for efficient exploration that effectively achieve the exploration-exploitation trade-off? How could incorporating human prior knowledge into RL accelerate the exploration process?
2. How can foundation models contribute to enhancing robotic manipulation? What roles can these models play in robot manipulation, and what are their limitations in practical applications?
3. What recent advancements have been made in integrating foundation models with reinforcement learning?

The remaining structure of this report is as follows:

- Chapter 2 introduces the basic concepts and methods of reinforcement learning, focusing on techniques that enhance exploration and sample efficiency.
- Chapter 3 explores the application of foundation models in robot manipulation, analyzing their functionalities and limitations.
- Chapter 4 reviews the current works in integrating foundation models with reinforcement learning.
- Chapter 5 discusses the existing limitations of this integrated framework and suggests directions for future research.

1.3. Methods

In this paper, we primarily retrieve literature using several methods. The first method involves identifying the most representative works in the field. These works typically have high reputations within the corresponding community and can be found through search engines or recommended by my supervisors. After locating these representative works, I use tools such as Google Scholar or Semantic Scholar to find other significant works cited in or referencing these papers. This approach allows me to construct a comprehensive network that maps the development of a specific field. The second method includes drawing insights from recent reviews or surveys in this field to compile a set of relevant literature. For example, I have acquired structured knowledge about reinforcement learning through the survey by [48] and OpenAI's Spinning Up [2]. For foundational models, I have consulted surveys by [33, 46]. These sources serve as a knowledge base, providing a clear categorization of methods and foundational knowledge. Utilizing these resources is beneficial for structuring the literature more systematically.

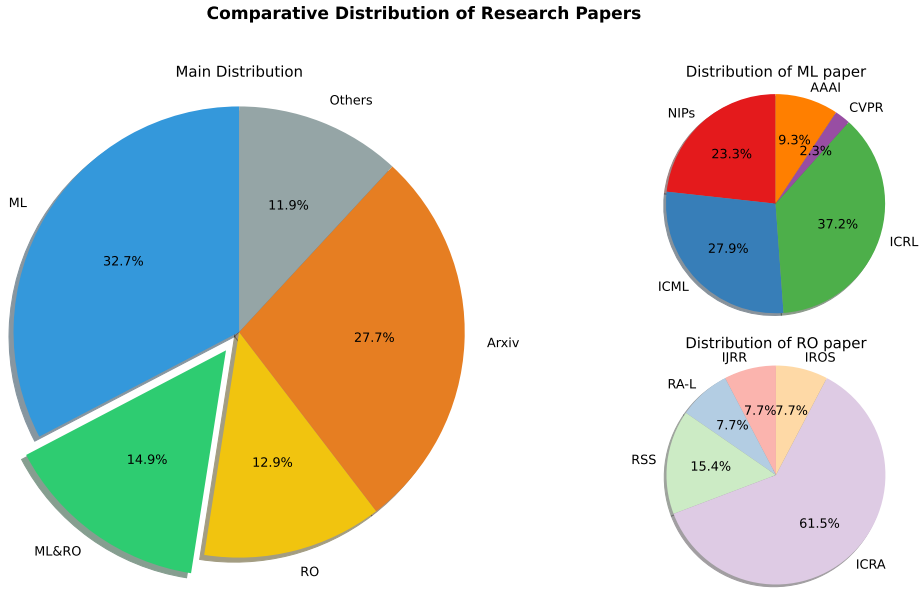


Figure 1.1: Distribution of literature

For the distribution of the literature, I mainly cite papers from the robotics and machine learning communities, as the research focuses on a combination of foundation models and robot learning. As shown in Figure 1.1, 32.7% of the literature originates from major machine learning conferences such as NIPS, ICML, ICLR, AAAI, and CVPR. Additionally, 14.9% of the literature comes from the CoRL conference, which focuses on both robotics and machine learning. 12.9% of the papers are from robotics journals and conferences, such as IJRR, RAL, ICRA, RSS, and IROS. Given the rapid evolution of foundation models, some researchers publish their preliminary findings on platforms like arXiv before formal publication. The papers cited are from the most prestigious conferences and journals in their fields, representing the forefront of development at the time and significantly contributing to shaping the perspective presented in this literature review.

2

Reinforcement Learning and Exploration Methods

Despite the promising potential of reinforcement learning for training control policies in robotics, training RL agents for robotic tasks, particularly long-horizon manipulation tasks, remains challenging due to the difficulties associated with exploration in continuous action spaces. Robots often need to explore vast action spaces for millions of steps, struggling to achieve a stable policy that consistently reaches the optimal state distribution. To address this, effective exploration algorithms are crucial for manipulation tasks.

This chapter is structured to provide a comprehensive overview of reinforcement learning and its applications in robotics. Section 2.1 outlines the problem formulation and the core concepts foundational to understanding reinforcement learning. Following this, Section 2.2 delves into the theoretical background of widely used RL methods, their learning frameworks, and exploration techniques. Section 2.3 focuses on methods designed to enhance RL exploration by encouraging novelty and managing uncertainty. In Section 2.4, we discuss how demonstrations, used as prior knowledge, can significantly augment RL exploration. Section 2.5 introduces hierarchical approaches that facilitate multi-level exploration, thereby improving the efficacy of reinforcement learning strategies. Section 2.6 provides a concise introduction to curriculum reinforcement learning (CRL). This approach strategically progresses from simpler to more complex tasks, making initial exploration more manageable and gradually increasing the challenge to improve learning effectiveness.

2.1. Reinforcement Learning Framework

Reinforcement learning (RL) constitutes an interdisciplinary domain within machine learning and optimal control, focusing on learning a policy to optimize cumulative rewards through trial and error [91]. In contrast to supervised learning, RL does not rely on labeled input/output pairs or expert knowledge. Instead, it learns from environmental reward signals, with the agent independently identifying an optimal control policy that maximizes long-term rewards.

This methodology is particularly effective in robotics, enabling robots to autonomously determine optimal behaviors through trial-and-error interactions with their environments [48].

2.1.1. Problem statement in RL

Markov Decision Process (MPD) [80] is the most commonly used problem formulation in reinforcement learning problems. The term "Markov Decision Process" highlights that the system adheres to the Markov property, meaning that transitions depend solely on the current state and the most recent action, without influence from any previous history. An MDP is a 5-tuple $\langle S, A, R, P, \rho_0 \rangle$, where:

- S is the set of states named *State Space*,
- A is the set of actions named *Action Space*,
- $R : S \times A \times S \to \mathbb{R}$ is the *Reward Function*, with $r_t = R(s_t, a_t, s_{t+1})$,
- $P : S \times A \to \mathcal{P}(s)$ is the transition probability function, with $P(s_{t+1}|s_t, a_t)$ being the distribution of s_{t+1} given state s_t and action a_t,

- ρ_0 is the state distribution at the start of each episode.

2.1.2. Policies

The objective of RL and MDP is to identify an appropriate policy π for decision-making. The function $\pi(a_t|s_t)$ that maps state s_t to the currently optimal action a_t.

To find the optimal policy π, it should maximize the cumulative reward function, representing the sum of rewards over an infinite horizon, which is mathematically expressed as:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

Here, $\gamma \in (0, 1)$ is a discount factor that moderates the importance of future rewards. The probability distribution of a trajectory under a given policy is defined by:

$$P(\tau \mid \pi) = p_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t).$$

The expected return of a policy π, denoted by $J(\pi)$, is calculated through:

$$J(\pi) = \int_{\tau} P(\tau \mid \pi)R(\tau) = E_{\tau\sim\pi}[R(\tau)].$$

To find the optimal policy π^*, we solve the following optimization problem:

$$\pi^* = \arg\max_{\pi} J(\pi)$$

where $\arg\max_{\pi}$ seeks the policy that yields the maximum expected return.

2.1.3. Value Functions

In some RL algorithms, agents are trained to learn a state-action value function, known as Q-value, rather than a direct policy. The "value" is defined as the expected return when starting from a given state or state-action pair and subsequently adhering to a specific policy indefinitely. This is mathematically represented as:

$$Q^{\pi}(s, a) = E_{\tau\sim\pi}[R(\tau) \mid s_0 = s, a_0 = a]$$

The optimal Action-Value function, which encapsulates the highest expected return obtainable from any state-action pair, can be defined as:

$$Q^*(s, a) = \max_{\pi} E_{\tau\sim\pi}[R(\tau) \mid s_0 = s, a_0 = a]$$

This formulation seeks to identify the best possible return that can be achieved under any policy, thereby guiding the development of optimal decision-making strategies in RL environments.

2.1.4. Challenges in RL

- **Curse of Dimensionality** The essence of RL lies in its exploration of the state-action pairs within the state and action spaces. As the number of dimensions increases, the requirement for data and computational resources grows exponentially in order to cover the entire state-action space comprehensively. However, these spaces can be extensive, particularly in robotics. For instance, the Franka Emika Panda Robot, which features 7 degrees of freedom in its joints, presents a significant challenge. To address the joint-torque control problem through RL, this robot operates within a state space of dimensions 2×7 and an action space of 7 dimensions. The complexity increases substantially when training visuomotor policies that handle image inputs as the state space expands dramatically.

  The exploration-exploitation trade-off is a core challenge in RL that becomes increasingly complex with dimensionality. Agents must continuously decide whether to exploit known rewarding actions or explore new state-action pairs that might yield greater rewards in the future. As the state-action space expands and becomes more intricate, effectively navigating this trade-off becomes crucial for the success of RL algorithms.

- **Goal specification and Reward** In RL, the reward function implicitly specifies the desired behavior, with the primary goal of RL algorithms being to maximize the accumulated long-term reward. However, defining an effective reward function can be surprisingly challenging in practice, especially in the context of robot reinforcement learning [48]. This complexity arises from ensuring that the reward function accurately reflects the intended tasks and goals without inadvertently encouraging undesirable behaviors.
- **Simulation and Reality Gap** Although the Sim2Real gap is not a focus of our paper, it is a notable challenge in robotic RL, describing the discrepancy between simulated environments and real-world conditions. This gap can lead to significant issues when deploying learned policy directly in physical systems due to differences in dynamics and inaccuracies in the simulation modeling. While simulation offers a controlled and cost-effective environment for training and testing, bridging this gap requires robust strategies, such as domain randomization, to ensure that models developed in simulation perform reliably in real-world scenarios.

2.2. RL Base Algorithms

2.2.1. Taxonomy of RL Algorithms

There are two primary approaches for representing and training agents in model-free reinforcement learning: policy optimization methods and value function methods, as well as a combination of both, known as actor-critic methods. These approaches each have distinct strategies for learning to solve reinforcement learning problems.

- **Policy Optimization Method**: These methods directly adjust the policy—the mapping from states to actions—to maximize the expected return. By iteratively improving the policy based on return feedback from the environment, these methods seek to enhance the decision-making process directly. Because they directly optimize the policy, which ultimately determines the agent's actions, so they tend to be more stable and reliable than value function methods.
- **Value Function Method**: In contrast, these methods focus on estimating the value of being in a given state or performing a specific action in a state. By learning these values, the agent can indirectly determine the best actions by selecting those that maximize the expected value. For learning the value function, Bellman equation [7] is proposed to update the value function by only using a state-action-reward tuple $\langle s, a, r, s' \rangle$.

  $$Q^*(s,a) = \mathbb{E}_{s' \sim P}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a') \right]$$

  Those Q functions can be updated through the Temporal-Difference (TD) method:

  $$Q'(s,a) = Q(s,a) + \alpha(R + Q(s',a') - Q(s,a))$$

  However, value function methods can lead to instability due to several potential failure modes, such as approximation errors, bootstrapping, and using off-policy data [91].
- **Actor-Critics Method**: This approach combines the strengths of both policy optimization and value function methods. The actor updates the policy based on the guidance of the critic, which evaluates the action taken by estimating the value function. This synergy allows for more stable and efficient learning by balancing direct policy updates with value estimation. Currently, most methods in the field, to some extent, follow the actor-critic framework, making it a widely adopted model in reinforcement learning.

In reinforcement learning, agents are updated through interactions with their environment, from which they receive feedback. There are primarily two approaches to handling past data. On-policy reinforcement learning algorithms do not utilize old data to update the agent. off-policy algorithms leverage batches of old data for training.

- **On-policy Method**: This method strictly uses data collected from the current policy's execution to make updates, ensuring that the learning process aligns directly with the policy's behavior. This approach is exemplified by algorithms like Policy Gradient, where updates are based on the action taken under the currently evaluated policy. Because these methods do not employ

old data, they exhibit lower sample efficiency. However, they directly optimize the most critical objective—policy performance. Mathematically, on-policy data is essential for calculating these updates, as it ensures the integrity and relevance of the information used to refine the policy.

- **off-policy Method**: Off-policy methods allow the agent to learn from experiences not generated by the current policy. This is achieved by storing past experiences in a replay buffer, which the algorithm can then use to sample from. This approach decouples the policy under which data is collected from the policy being improved. However, a significant challenge with this method is that there are no assurances that effectively satisfying Bellman's equations will result in optimal policy performance. While empirical results can sometimes be great, offering impressive sample efficiency, the lack of guaranteed performance improvements renders these algorithms potentially unstable.
- **offline Method**: Offline RL operates entirely on a pre-collected dataset of experiences without further interaction with the environment during learning. This method is particularly advantageous when online data collection is risky, expensive, or impractical. Offline RL aims to derive the best possible policy by utilizing a comprehensive static dataset. However, the reliance on a fixed dataset poses unique challenges, such as distribution shift and overfitting.

2.2.2. Core RL algorithms

In this section, we intend to introduce the core algorithms in the RL community. We mainly focus on their theory background and their design to achieve exploration and exploitation trade-offs and higher data efficiency.

2.2.2.1.On-policy method

- **Vanilla Policy Gradient**: The concept behind training with vanilla policy gradients (VPG) [91] is to increase the probabilities of actions that result in higher returns. This is achieved by defining the gradient as follows:

$$\nabla_\theta J(\pi_\theta) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)$$

  This equation essentially treats policy training as a weighted classification problem, where the weights are provided by the episode return function $R(\tau)$.

  Exploration in VPG is facilitated by its inherent nature as a stochastic policy. As training progresses, the policy gradually learns to become more deterministic, increasingly outputting actions that offer higher returns.

- **A2C**: The key distinction between Advantage Actor Critics (A2C) [65] and VPG is that A2C incorporates an advantage function through a critic network to mitigate the variance associated with the return estimates. The advantage function, defined as $A(s,a) = Q(s,a) - V(s)$, measures how much better taking a particular action is over the average.

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A^{\pi_\theta}(s_t, a_t) \right]$$

  This formula underscores how A2C updates policy parameters by focusing on the most advantageous actions, improving efficiency and convergence over traditional VPG.

- **PPO**: Trust region methods [85, 84] aim to update policies by taking the largest step possible to improve performance without stepping so far that we accidentally cause performance collapse. The problem of the trust region policy gradient method is defined as:

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}(\theta_k, \theta) \quad \text{s.t.} \quad D_{KL}(\theta \parallel \theta_k) \leq \delta$$

  where the policy loss is calculated through:

$$\mathcal{L}(\theta_k, \theta) = \mathbb{E}_{s,a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s,a) \right]$$

Proximal Policy gradient (PPO) [84] uses a few other tricks to keep new policies close to old which is significantly simpler to implement than TRPO [85].

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a\sim\pi_{\theta_k}} [L(s, a, \theta_k, \theta)]$$

The loss of PPO can be defined as:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right)$$

Similar to VPG and A2C, PPO trains a **stochastic policy** that can achieve exploration and exploitation trade-offs.

2.2.2.2.Off-policy method

- **DQN and DDPG**: Deep Q-Networks (DQN) [66, 31] represents a significant breakthrough in reinforcement learning, combining classical Q-learning principles with the power of deep neural networks as Q value approximation, which efficiently handle environments with high-dimensional state spaces. Compared to classic Q-learning, DQN uses a deep neural network to approximate the Q-function instead of using a tabular approach to store Q-values.

  One of the key innovations in DQN is the use of experience replay. As the agent interacts with the environment, it stores transitions in a replay buffer. These transitions are tuples of (current state, action, reward, next state). The replay buffer then randomly samples mini-batches of these transitions to train the network.

  $$L(\phi, D) = \mathbb{E}_{(s,a,r,s',d)\sim D} \left[\left(Q_\phi(s, a) - (r + \gamma(1 - d) \max_{a'} Q_\phi(s', a')) \right)^2 \right]$$

  DQN do not have an explicit policy network, but they use maximum action in the discrete action space:

  $$a^*(s) = \arg\max_a Q^*(s, a).$$

  .

  To balance exploration and exploitation, DQN employs an **epsilon-greedy** strategy. With a probability of ϵ, the agent chooses a random action, promoting exploration of the environment. With a probability of $1 - \epsilon$, the agent selects the action with the highest predicted Q-value from the network, exploiting its current knowledge

  Deep deterministic policy gradient (DDPG) [55] can be considered as DQN for continuous action space. It explicitly introduces a policy network for continuous action space, which results in changes in the network update of critics.

  $$L(\phi, D) = \mathbb{E}_{(s,a,r,s',d)\sim D} \left[\left(Q_\phi(s, a) - (r + \gamma(1 - d) Q_\phi(s', \mu_\theta(s'))) \right)^2 \right]$$

  The gradient of the policy's performance is estimated using the critic's Q-value output, essentially performing gradient ascent on the actor's parameters.

  As DDPG is a deterministic algorithm, it would probably not try a wide enough variety of actions initially. To make deterministic policy explore well, action noise is introduced to improve the variety for trajectory at training time.

- **TD3**: Twin Delayed Deep Deterministic Policy Gradient (TD3) [25] is an enhancement over the DDPG algorithm, designed to address overestimations in Q-value calculations and to prevent the exploitation of Q-function estimates. TD3 incorporates three key tricks to mitigate these issues: **target policy smoothing**, clipped double-Q learning [95], and delayed policy updates.

  Like DDPG, TD3 also adds noise to the deterministic policy as exploration. TD3 also adds noise to target action for policy updates, which avoids the exploitation of the incorrect sharp peak in the critics' network. It helps to achieve a balance between exploration and exploitation.

- **SAC**: Soft Actor Critics (SAC) [30] is an RL algorithm incorporating entropy regularization into the stochastic policy to enhance exploration. This approach is quantified by the entropy of the policy:

$$H(P) = \mathbb{E}_{x\sim P}[-\log P(x)].$$

The objective of SAC is to develop a policy that not only maximizes expected return but also maintains a high level of stochasticity facilitated by entropy. The degree of exploration is controlled by the entropy coefficient α, where a higher α promotes more exploratory behavior:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot|s_t))\right)\right].$$

The inherent randomness introduced by this policy aims to enhance exploration and make the policy more robust to noise. Also, to avoid exploiting the incorrect sharp peak in the critics' network, SAC employs a clipped double-Q learning strategy to prevent value overestimation.

$$y(r, s', s, d) = r + \gamma(1-d)\left(\min_{j=1,2} Q_{\theta_j}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

Table 2.1: Comparison of Reinforcement Learning Algorithms and Exploration Methods

Method	Action Space	Policy Stochasticity	Tricks for exploration and data efficiency	Explicit Exploration Method
VPG	Both	Stochastic	No	No
A2C	Both	Stochastic	No	No
PPO	Both	Stochastic	No	No
DQN	Discrete	Deterministic	Experience replay	Epsilon-greedy
DDPG	Continuous	Deterministic	Experience replay, Added noise to actions	No
TD3	Continuous	Deterministic	Experience replay, Added noise to actions, Clipped double-Q, Target policy smoothing	No
SAC	Continuous	Stochastic	Experience replay, Clipped double-Q	Entropy regularization

This section provides an overview of fundamental reinforcement learning algorithms and their approaches to balancing exploration and exploitation, as detailed in Table 2.1.

Most on-policy RL algorithms, such as VPG, A2C, and PPO, naturally incorporate exploration through their inherently stochastic policies. This intrinsic randomness serves as a mechanism for exploration without explicit exploratory actions. Off-policy RL methods, including DQN, DDPG, and TD3, utilize a replay buffer to store past transition tuples, significantly enhancing training sample efficiency. However, these methods can experience unstable training and policy behavior. To encourage exploration, these algorithms often add randomness to the policy, enabling more diverse rollout trajectories. This is typically achieved through strategies like random discrete action selection in DQN or action noise in DDPG and TD3. Also, the SAC algorithm extends these concepts by incorporating entropy regularization directly into the policy. This technique encourages a richer on-policy data distribution and balances exploration and exploitation, aiming to increase the overall robustness and effectiveness of the policy learning process.

2.3. Exploration Method In Reinforcement Learning

2.3.1. Count-Based Exploration

There is a concept called "optimism in the face of uncertainty" (OFU), widely used as a heuristic in sequential decision-making problems. This concept operates on the principle that decision-makers should prefer actions with uncertain outcomes when the potential for positive results exists, especially when information is incomplete.

A key method employing this heuristic is the Upper Confidence Bound (UCB) algorithm [4], which balances exploration and exploitation by prioritizing actions that have either been less explored or shown promising results. When selecting the next action, the agent considers not only the cumulative rewards but also a reward for optimism of uncertainty, primarily determined by the frequency of visits to each action. The selection rule is represented by the formula:

$$a_t^* := \arg\max_{a \in A} \left(Q(s_t, a) + C \sqrt{\frac{\log \sum_{a'} N(s_t, a')}{N(s_t, a)}} \right)$$

This foundational approach has paved the way for further developments in count-based exploration techniques.

However, traditional methods like UCB function effectively in environments with discrete state and action spaces. In continuous or complex environments, directly counting the $N(s_t, a)$ is infeasible. A study [6] proposes an algorithm that derives a pseudo-count from an arbitrary density model to estimate uncertainty, allowing count-based exploration strategies to be adapted to non-tabular settings. Building on this, [75] improved this approach by integrating PixelCNN pseudo-counts with Monte Carlo updates, significantly advancing exploration capabilities in sparse scenarios. Additionally, authors in [62] developed a technique that rewards agents for exploring transformed feature spaces instead of raw state spaces. #Exploration [92] implemented state mapping to hash codes, enabling state occurrence counting via a learned hash table and applying traditional count-based exploration rewards to modern and complex environments.

2.3.2. Novelty Motivated Exploration

State counts are not the only indicator of an RL agent's knowledge about specific states. There are various other techniques available to motivate exploration and measure uncertainty.

System dynamics serve as a measure of uncertainty in [90]. The authors propose a method that utilizes prediction errors from states and actions as bonuses in the reward function. In this context, a larger prediction error indicates greater uncertainty about the state. This scalable and efficient approach makes it well-suited for exploration bonuses in tasks involving complex, high-dimensional state spaces. Building on this concept, inverse dynamics has also been utilized to quantify state uncertainty, incentivizing the agent for exploration in [77].

However, these methods that rely on prediction errors during exploration often gravitate towards transitions with high stochasticity. To address this, the authors of RND [12] proposed using the error of a neural network predicting features given by a fixed randomly initialized neural network as an exploration bonus instead of focusing solely on error in transitions. Despite its strengths, this method can sometimes concentrate exploration too narrowly, neglecting other potential areas of interest. Addressing this limitation, NovelD [109] introduces a straightforward solution by applying approximately equal weighting to all novel areas. While still employing RND to determine novelty, NovelD allocates the calculated difference as an intrinsic reward, thus prioritizing unexplored boundary states. This approach results in more efficient and extensive exploration patterns, effectively broadening the scope of discovery within the environment.

In reinforcement learning, count-based and curiosity-driven methods have demonstrated effectiveness primarily in structured environments such as mazes or games like Montezuma's Revenge, where exploring a broad expanse of the state space is integral to success. These tasks explicitly require agents to traverse and explore themselves with nearly all possible states to complete the objectives effectively. However, the configuration space in robotics applications often lacks a direct correlation with successful outcomes. Here, novelty or curiosity-driven exploration does not inherently guarantee meaningful or purposeful exploration. The challenge lies in adapting these exploration strategies to ensure relevance and utility in the varied and often unstructured contexts encountered in robotics.

2.4. Efficient Reinforcement Learning with Demonstration

In robot manipulation tasks, the actions performed by a robot often convey semantic information, such as "pick up the block" or "reach a position." For RL agents to explore these tasks effectively, it is crucial that they grasp the meanings of these actions, rather than simply traversing every possible

joint or end-effector state. Traditional exploration methods often fail to incorporate this semantic knowledge into the RL agent's learning process. Demonstrations, which are robot trajectories collected by human experts, serve as a bridge transferring human prior knowledge to machine learning agents, employing techniques such as behavior cloning or inverse reinforcement learning (IRL). Since 2017, several researches has focused on utilizing demonstrations and offline data to reduce the exploration period and accelerate the training of RL agents. This approach allows agents to rapidly assimilate complex skills by leveraging human insights.

DDPGfd [97] is the first framework that integrates demonstrations into the replay buffer of an off-policy reinforcement learning algorithm. To effectively balance online transitions with offline demonstrations during training, it incorporates a prioritized replay buffer that samples transitions based on their temporal-difference (TD) error. Building upon the foundational DDPG model [55], this work also introduces a combined loss function for the critic network, which encompasses both 1-step and multi-step returns. This innovative framework has enabled reinforcement learning to successfully tackle robot insertion tasks using just 100 demonstrations, significantly enhancing the learning efficiency. DQfd [32] is an approach similar to DDPGfd, but it builds upon the DQN algorithm [66]. Like its counterpart, DQfd incorporates demonstrations into the replay buffer to facilitate learning from offline data. This method enhances training efficiency for Atari games, which feature discrete action spaces, by employing a margin loss. This loss function specifically encourages expert actions to have higher Q-values compared to all other actions.

A concurrent study [70] focuses on addressing the exploration challenges in long-horizon, multi-step robotics tasks involving continuous control, such as stacking blocks with a robot arm. This paper systematically explores various methods to integrate demonstrations with the DDPG reinforcement learning technique to enhance training efficiency and performance. These methods include the introduction of a demonstration buffer, the incorporation of behavior cloning loss to regulate the actor network, the use of hindsight replay buffers, and the strategy of resetting to demonstration states.

Prior works have primarily focused on utilizing demonstrations to enhance sample efficiency. In contrast, AWAC [69] shifts the focus to another aspect of integrating offline data with online processes. Specifically, AWAC emphasizes using online RL exploration to fine-tune sub-optimal data from offline sources. While previous methods [32, 82, 70] have employed behavior cloning loss, this can sometimes limit the policy's potential to outperform the demonstrations due to their inherent suboptimality. AWAC addresses this limitation by introducing a maximum likelihood loss function(behavior cloning loss) weighted by advantages function.

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a\sim\beta}\left[\log \pi_\theta(a \mid s) \exp\left(-\frac{1}{\lambda} A^{\pi_k}(s,a)\right)\right]$$

This adjustment allows AWAC to leverage online exploration effectively to fine-tune policies based on offline data, bridging the gap between the sub-optimal behaviors and optimal policy learning.

RLPD [5] introduces a method that integrates efficient online RL techniques with offline demonstrations. Drawing inspiration from REDQ [14], RLPD incorporates several elements to expedite online learning, including a high Update-to-Data (UTD) ratio and ensemble critics. The UTD ratio, which is the ratio of policy updates to environment steps, enhances data efficiency, particularly for off-policy methods that already include offline demonstrations in the replay buffer. However, it risks overfitting to data collected in the early stages. To mitigate this overfitting, ensemble critics are employed through randomized ensemble distillation, which also helps prevent catastrophic overestimation—a phenomenon where the Q-function overestimates the Q-value of out-of-distribution samples, leading to divergence in the critics. Furthermore, the paper introduces additional elements such as a balanced ratio of online-to-offline data in each batch and layer normalization to better integrate offline data. These innovations have enabled RLPD to achieve state-of-the-art performance in the current RL community for using offline data with online fine-tuning.

Some research [51, 71]focuses on using online exploration to refine and enhance offline RL policies. Traditional offline RL methods have often struggled with online fine-tuning due to state-action distribution shifts, which can lead to severe bootstrap errors. This issue can be addressed through approaches such as Balanced Replay and Pessimistic Q-Ensemble [51], and Q-value calibration [71]. These techniques mitigate key challenges associated with distribution shifts and utilize demonstrations more effectively and reduce the necessary exploration phase.

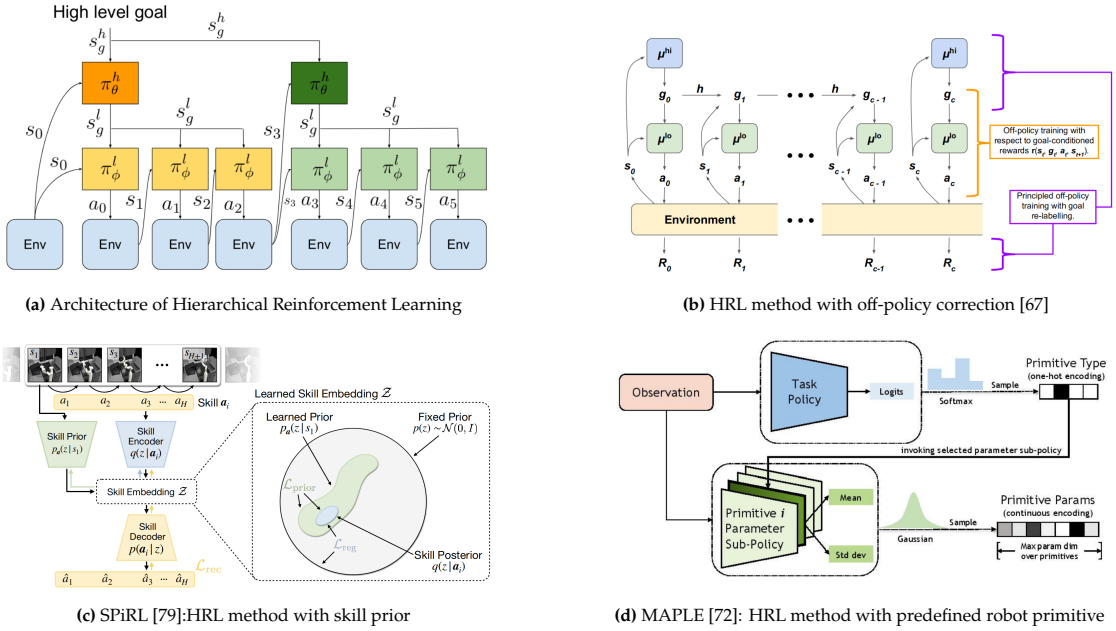
Table 2.2: Efficient RL methods with demonstration

Method	Basic RL Algorithm	Replay Buffer Design	Additional Loss	Other Designs
DDPGfD [97]	DDPG	prioritized-replay	L2	-
DQNfD [32]	DQN	prioritized-replay	L2 & BC	-
Nair et al., [70]	DDPG	hindsight-replay	BC	reset to demonstration state
AWAC [69]	Actor Critics	-	Advantage weighted BC	-
RLPD [5]	SAC	with 50% offline data ratio	-	Layer normalization, ensemble critics

However, a significant assumption underlying these works is that demonstrations are readily accessible, which may not always be the case. Additionally, the quality of these demonstrations greatly impacts the efficiency of the learning process. Demonstrations from experts can significantly accelerate learning and reduce the need for exploration, while demonstrations of lower quality may lead to inefficient learning processes and necessitate additional exploration. Moreover, if the demonstrations do not cover specific meaningful regions of the state-action space, the RL algorithm must still undertake substantial exploration on its own. This dependence on the quality and comprehensiveness of demonstrations imposes significant limitations on methods that integrate reinforcement learning with demonstrations.

2.5. Goal-based Method For Reinforcement Learning

When faced with large state and action spaces and extended task horizons, exploration using standard RL approaches becomes challenging. Hierarchical Reinforcement Learning (HRL) addresses this by decomposing complex tasks into simpler subtasks through a hierarchy of policies, each learned via reinforcement learning, as depicted in Figure 2.1a. The advantages of this hierarchical structure are thoroughly proven and analyzed in [68]. The study concludes that hierarchical modeling facilitates RL training and exploration within more semantically meaningful action spaces, enhancing both efficiency and effectiveness.



(a) Architecture of Hierarchical Reinforcement Learning

(b) HRL method with off-policy correction [67]

(c) SPiRL [79]:HRL method with skill prior

(d) MAPLE [72]: HRL method with predefined robot primitive

Figure 2.1: A illustration of methods in HRL

Early works such as FeUdal [98] and h-DQN [49] integrate hierarchical modeling in modern deep

reinforcement learning, where a high-level agent specifies goals and a low-level agent works to achieve them. The high-level policy is trained using extrinsic rewards, while the low-level controller is trained with intrinsic rewards focused on goal attainment. Hierarchical modeling thus provides an effective framework for exploration in complex environments, such as tasks with long horizons.

However, these methods often require careful, task-specific design, which can limit their generality. Additionally, changes in the behavior of the lower-level policy can create non-stationary issues for the higher-level policy, reducing stability during off-policy training. As illustrated in Figure 2.1b, HIRO [67] addresses these challenges by implementing off-policy corrections. It involves relabeling high-level transitions with alternative high-level actions that maximize the probability of achieving the set goals. Furthermore, HIRO directly uses the state as the goal and defines the reward function as the distance to this goal, enhancing the method's applicability across a broader range of scenarios.

One approach, presented in [28], integrates HRL with unstructured demonstrations. This method comprises an imitation learning stage that develops goal-conditioned hierarchical policies, followed by a reinforcement learning phase that fine-tunes these policies for enhanced task performance. Similarly, SPiRL [79], shown in Figure 2.1c, leverages offline data within an HRL framework. It focuses on extracting skill embeddings and skill priors from offline data, subsequently training an HRL policy guided by these priors. This approach eliminates the need to learn low-level behaviors directly and guides the policy training process through pre-learned skills.

Robot manipulation is a field where actions carry significant semantic meanings. MAPLE [72] introduces a method that utilizes predefined primitives, such as Reaching, Grasping, Pushing, Releasing, and Atomic actions, as high-level representations, as illustrated in Figure 2.1d. To effectively utilize these heterogeneous primitives, a hierarchical policy is developed, which organizes the primitives and facilitates their execution with specific input parameters. Experiments conducted in tasks such as Pick-Place and Assembly demonstrate that these behavior primitives can substantially enhance exploration efficiency.

2.6. Curriculum RL

Curriculum learning is a method that starts with easier tasks to facilitate the learning of more difficult tasks, thereby enhancing the efficiency of RL, particularly in robot manipulation. One approach within curriculum learning involves generating start-state distributions. For instance, a reverse exploration method proposed by [23] sets the start state of an episode by beginning at the goal state, thereby creating a more generalized start distribution. Similarly, BaRC [41] uses a given physical model to initiate from the goal state and spread the start state distribution backward. These methods employ a reverse search to select goal states, starting with the easiest states and progressing to more difficult ones, simplifying the achievement of goals. Additionally, another method by [22] uses a generative model to create goal states for curriculum learning, enhancing adaptability and scalability.

Furthermore, CHER [21] introduces a method that applies curriculum learning concepts to training data. It generates goals in the hindsight replay buffer as part of the curriculum, balancing curiosity and proximity. These goals are then utilized to recalculate rewards in the hindsight replay buffer. Another approach adjusts the tolerance levels of tasks from easy to difficult as part of the curriculum. PCCL [59] modifies the precision of a success detector, starting with easier criteria and gradually increasing difficulty throughout the training process.

While curriculum learning in reinforcement learning significantly accelerates task learning and enhances model adaptability, a key limitation is its dependency on accurately defining task difficulty and sequence, which can vary widely between different environments and learning objectives.

2.7. Conclusion

This chapter has introduced the foundational concepts and methodologies of RL, focusing on enhancing exploration and sample efficiency. Novelty-driven and count-based methods, discussed in Section 2.3, have shown promising results in exploring state spaces in games and mazes. However, their nature may not align well with manipulation tasks, where most states in the robot action space may be meaningless.

Both demonstrations and hierarchical modeling have significantly improved sample efficiency within robotics. To some extent, each method integrates prior knowledge into robotic systems. Demonstrations serve as an intermediate for transferring human knowledge to RL agents, effectively bridging the gap between human expertise and machine learning. Hierarchical modeling reflects the human cognitive

model in structuring robot learning for long-horizon manipulation tasks, enabling robots to learn what to do (high-level) and how to do it (low-level). Exploring both high-level and low-level aspects can lead to a more efficient exploration process.

Despite the effectiveness of these methods, questions remain about their sufficiency. The aforementioned approaches still require millions of steps to converge or hundreds of pre-collected demonstrations by human experts. This raises an important question: Is there another source of prior knowledge that could accelerate RL training more effectively?

3

Foundation Models for Manipulation

In this chapter, we explore the application of foundation models in robotic manipulation tasks. We concentrate on the "reasoning capability" of these models, particularly how they perform in robot manipulation tasks under zero-shot or few-shot conditions.

In Section 3.1, we introduce core concepts and architectures of foundation models. Section 3.2 analyzes the use of these models for high-level planning in robotic tasks. In Section 3.3, we review current literature that leverages vision-language models (VLM) for perception in robot tasks, focusing on scenarios involving unseen environments. Finally, Section 3.4 discusses methods that incorporate feedback to refine and correct plans generated by foundation models, enhancing their applicability in dynamic settings.

3.1. Foundation Models Introduction

Recent advances in natural language processing (NLP) and machine learning, particularly the development of the Transformer architecture [96], emerge a new category of models known as foundation models [8]. These models, such as Large Language Models (LLMs) and Vision-Language Models (VLMs), are characterized by their extensive training on broad datasets. This extensive training enables them to perform effectively across various tasks and modalities, even those not encountered during their training.

This versatility enables foundation models to be powerful tools in tackling diverse challenges across various fields, including robotics. In robotics, these models can be integrated as key components for tasks such as perception, planning, and control, enhancing the robot's ability to interpret and interact with its environment effectively.

3.1.1. LLMs

Large language models refer to large deep neural networks that typically incorporate multiple transformer blocks in their architecture. These models are trained auto-regressively to predict the next token in a sequence. Some of the most commonly used LLMs are GPT-3 [11] and GPT-4 [1], which have demonstrated remarkable effectiveness across various language processing tasks. Leveraging extensive text datasets in the training phase, these models are able to generate coherent and contextually relevant text for tasks such as text completion, question answering, and even code generation. GPT-4 Vision extends this functionality to include vision input modalities, enhancing the model's versatility and applicability to multimodal tasks.

Furthermore, GPT and other LLMs such as Llama2 [93], Liama3 [36] and Mistral [43] can be prompt through Chain of Thought (CoT) [101] technique. This method enables the models to exhibit more complex inference capabilities by iterating through intermediate reasoning steps while solving a problem, thereby improving their problem-solving processes and outcomes.

3.1.2. VLMs

Language input alone often falls short in solving complex tasks, as it may not capture detailed information comprehensively. In contrast, vision modalities can provide a rich description of the environment, encapsulating extensive details within images. Vision-language Models (VLMs) are trained with

Internet-scale massive datasets and designed to integrate and interpret visual and linguistic information. These models effectively bridge the gap between vision and language, enabling them to perform complex tasks that require an understanding of multimodal inputs.

CLIP [81] leverages contrastive training to correlate visual inputs with language using a 400 million image-text pairs dataset. It employs separate encoders for text and images, allowing each modality to be encoded independently. CLIP effectively maps both modalities into a shared latent space by computing the cosine similarity between text and image representations. In this space, texts and images with similar content are positioned closer together, facilitating a more aligned understanding across visual and textual data.

Several studies have successfully extended the capabilities of VLMs to tasks such as open-language object detection and localization. Notable examples include VILD [26], OWL-ViT v1 [64], OWL-ViT v2 [63] and DinoV2 [74]. Those models facilitate more complex and flexible open-language object detection tasks, which could be valuable for robotics perception modules.

Segment Anything [47] is a Large Vision Model specialized in image segmentation tasks. It is trained on the largest segmentation dataset, which contains 1 billion masks on 11 million images. This model can transfer knowledge zero-shot to unseen image distributions through prompts, enabling it to adapt to new visual contexts effectively.

3.2. Foundation Models as Planning

Foundation models have proven their capability in reasoning and contextual generalization, making them increasingly popular in the robotics community for high-level (task-level) planning. Large Language Models (LLMs) decompose complex tasks into discrete, primitive-level action steps in such applications. A notable example from SayCan [3] illustrates this application effectively: consider a scenario where a human gives the instruction, "I spilled my drink, can you help?" The LLM would break down this task into the following steps:

1. find a sponge
2. pick up the sponge
3. come to you
4. put down the sponge
5. done

These actions are subsequently executed by a low-level controller, which could involve techniques such as pretrained behavior cloning policies, pretrained RL policies, or model predictive control, among others. This approach ensures that the high-level commands decomposed by the foundation models are translated into precise, real-world movements and tasks handled efficiently by the control systems of various embodiments.

However, one of the main concerns is that large language models may lack sufficient insights into real-world environments and embodiments when addressing high-level planning problems. It is challenging for a language model to fully comprehend complex geometries, action feasibility, and embodiment kinematics. These gaps in understanding can lead to failures in executing high-level plans effectively, highlighting the need for more integrated and context-aware approaches in robotics.

3.2.1. Real World Grounding

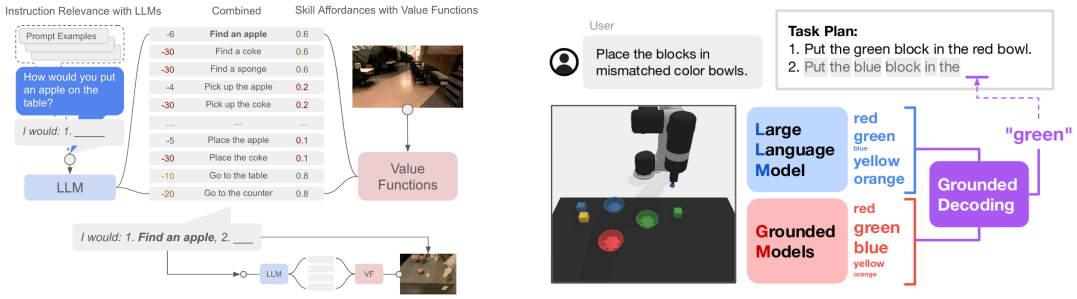
Grounding is a crucial concept in integrating foundation models with robotics. It refers to the ability of a system to associate contextual meaning with signals or symbols. Specifically, grounding involves aligning the abstract knowledge possessed by foundation models with real-world environments and robotic systems. This alignment ensures that language-driven decisions correspond meaningfully with physical actions and environmental contexts, enhancing the relevance and effectiveness of robotic operations based on language instructions.

- **Pretrained Skills/Affordances as grounding**:

  SayCan [3], as depicted in Figure 3.1a utilizes pretrained skills as a means of grounding in the real world, which constrains the model to propose natural language actions that are both feasible ("Can") and contextually appropriate ("Say"). To achieve this, the authors integrate an affordance

score—which reflects the success rate of a specific pretrained skill—with the language scores. Ultimately, the skill with the highest combined score is selected for execution by the pretrained policy. This planning method is iteratively run until the task is completed, ensuring each step is practical and relevant to the current context.

Grounding Decoding [37] also integrates language model planning with a grounded pretrained affordance model. Unlike SayCan, which selects skills for task-level planning, Grounding Decoding operates at the token level by jointly decoding the next word in the sentence from both LLM and the Grounded Models, as illustrated in Figure 3.1b. This approach allows the system to engage in open language task planning without the constraints of predefined skill libraries, thereby enhancing flexibility and applicability to a broader range of scenarios.



(a) Saycan framework [3]

(b) Grounded decoding framework [37]

Figure 3.1: A illustration of methods using Affordance as world grounding

However, most recent works that combine LLM planning with affordance grounding require a set of low-level language-conditioned policies and a grounded model, which are not always accessible. This dependency can pose challenges in implementing such systems across different applications and environments.

- **Feedback as grounding**:

  Inner Monologue [38] explores how to utilize environmental feedback to enable an LLM to develop an inner monologue for reasoning over various types of feedback, including success detection, scene description, and human instruction.

  Recent research investigates how human feedback can refine plans generated by LLMs, encompassing several approaches: DROC [107] focuses on distilling knowledge from human feedback; CoPAL [44] develops a system architecture that handles corrections at different levels; and REFLECT [58] introduces a framework that prompts LLMs to reason about failures using a hierarchical summary of the robot's past experiences. These studies enhance the adaptability and effectiveness of LLMs in dynamic and interactive environments.

  However, relying solely on feedback can not ensure successful planning, particularly in complex environments where unpredictable variables can affect outcomes.

- **Perception as grounding**:

  VLMs are valuable tools for the Perception module in robotics, serving as effective scene descriptors. Socratic Models [105] aims to leverage the combined knowledge and capacities from different modalities—vision and language—to aid in robot planning tasks. This integration allows the utilization of commonsense knowledge across different domains. VLMs are employed for open language object detection, translating the results into language to provide an environmental grounding for LLM planning. Instead of the language descriptions from VLMs, Voxposer [40] uses VLMs to construct a 3D value map that serves as the environmental grounding. The robot then performs model predictive control based on this cost map.

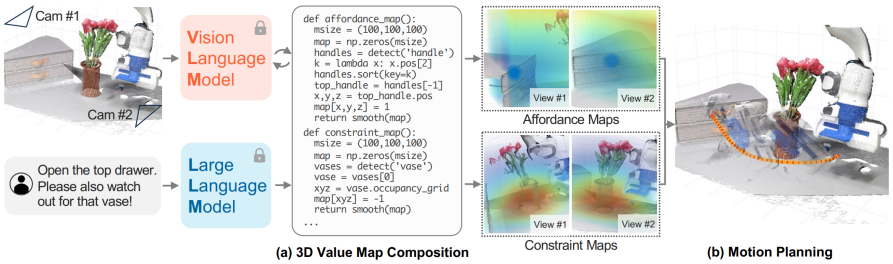


Figure 3.2: Voxposer [40]: a framework that utilizes LLM to generate cost map for planning

VILA [34] takes a direct approach by using LLMs with vision modalities (e.g., GPT-4 Vision) for planning without combining with a VLM. GPT-4 Vision can process visual inputs and effectively "observe" the environment. This approach integrates perceptual data into the reasoning and planning process and enhances the model's ability to apply commonsense knowledge in visual contexts, demonstrating its superiority over traditional LLM-based planners.

- **Code as grounding**:

  Using LLMs to generate plans represents an explicit method of task-level planning. Still, there are also more implicit methods that enable LLMs to undertake planning, such as through code generation. Code as Policies (CAP) [54] introduces an approach for robotic planning by generating code. When provided with several example language commands formatted as comments, followed by corresponding policy code via few-shot prompting, LLMs can process new commands and autonomously re-compose API to generate robot-centric policy code accordingly.

  This method of utilizing LLMs goes beyond simply planning a sequence of skills. The ability of LLMs to write code allows them to handle spatial relationships (e.g., "move the apple a bit to the left"), incorporate commonsense priors in control (e.g., "move faster", "push harder"), and operate effectively even with a limited skill library. This approach broadens the applicability of LLMs in complex environments and enhances their utility in precise and context-sensitive tasks.

  ProgPrompt [88] utilizes a code-completion LLM to generate robotic plan programs, enhancing situated awareness in LLM-based robot task planning. The authors have crafted a programmatic LLM prompt structure to facilitate plan generation that is adaptable across various environments, robot capabilities, and tasks. A key advantage of using code is its ability to effectively handle state feedback from the environment. For instance, when given the instruction "make dinner," the availability of ingredients like chicken, soda, or pickles in the refrigerator may vary, which traditional LLM-based task planning methods might struggle to accommodate. However, the code-generated by ProgPrompt can dynamically adjust to these variations by incorporating real-time state information, ensuring more accurate and context-aware task execution.

  Instruct2Act [35] leverages an LLM to generate Python programs constituting a comprehensive perception, planning, and action loop for robotic tasks. Within the perception module, pre-defined APIs provide access to multiple foundation models, such as SAM [47] and CLIP [81]. This framework can derive robotic plan codes that integrate code from perception foundation models, pretrained action policies, and hard-coded APIs and IO operations.

  Compared to methods that prompt LLMs for high-level plans, code-based LLM methods can handle a wider variety of environments and instructions without the constraints of a predefined skill library. However, regarding execution accuracy, current code generation methods are typically limited to simple tasks, such as pick-and-place operations. It is challenging for LLM-generated code policies to execute complex tasks with high precision without the support of pretrained skills.

3.2.2. LLM with TAMP

Some studies have utilized LLMs for robot task-level planning, such as SayCan [3] and Inner Monologue [38]. However, language models alone cannot reliably solve long-horizon robot planning problems because language lacks complete environmental information. Additionally, by only myopically executing

the next skill at each timestep, they may fail to account for geometric dependencies that span the entirety of a skill sequence. Recently, numerous works have aimed to integrate the "commonsense knowledge" embedded in LLMs with classic task and motion planning (TAMP). This integration can enhance the feasibility and efficiency of handling symbolic and geometric relations.

Some research has used planning frameworks to solve long-horizon reasoning with LLMs and Q-functions. Text2Motion [56] uses feasibility heuristics encoded in Q-functions of a library of skills to guide task planning with LLMs. They proposed a method to combine shooting-based (planning full sequences) and search-based (one-by-one in a greedy manner) planning strategies to construct geometrically feasible plans for long-horizon tasks.

LLM+P [57] represents the first framework to integrate the strengths of classical planners into LLMs. Instead of using LLMs as planners, some literatures employ LLMs to generate a PDDL [24] description of the given problem and then translate the plan back into natural language. This framework has demonstrated superior feasibility and spatial reasoning in robotics tasks compared to LLM-AS-P. AutoTAMP [15] translates natural language task descriptions into a TAMP task representation that can then be solved with task and motion planning. This method also re-prompts the LLM autoregressively to detect and correct both syntactic and semantic errors, significantly improving task completion rates.

While PDDL plans are symbolic, a symbolically correct skill sequence may fail during execution due to the robot's kinematic constraints or geometric dependencies across the skill sequence. To address these issues, some research combines planners with LLM planning. LLM^3 [99] use a motion planner to rollout the result from LLM. Feedback from the classic motion planner, such as collision or unreachable states, serves as implicit heuristics for the LLM planner, guiding it to replan based on this feedback.

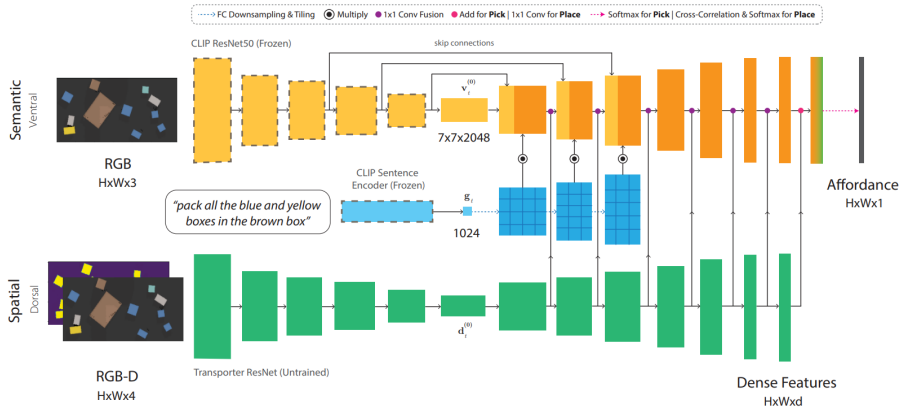
The results generated by LLMs are not necessarily optimal, necessitating the use of TAMP planners for feasible and efficient task-motion plans. LLM-GROP [18] prompts LLMs to extract commonsense knowledge about semantically valid object configurations and integrates this with a task and motion planner to generalize to varying scene geometries. After LLM generates symbolic and geometric spatial relationships between the tableware objects, Task and Motion Planner GROP [110] generates the optimal plan for the robot to execute based on the information provided by the LLM.

All the works mentioned above demonstrate improved performance by integrating planning structures, symbolic planning, or motion planning with LLMs. Introducing these planning modules helps LLMs achieve more accurate grounding in logical and spatial relationships between objects. Compared to traditional planning frameworks, LLMs enable handling tasks with open language inputs without the need for manual setup. However, it is noteworthy that although these methods focus on robot manipulation as the environmental setting, none address contact-reach tasks, a domain where classic planning frameworks also face challenges. Contact-reach tasks commonly require robots to learn how to make contact with objects using machine learning techniques such as reinforcement learning and imitation learning.

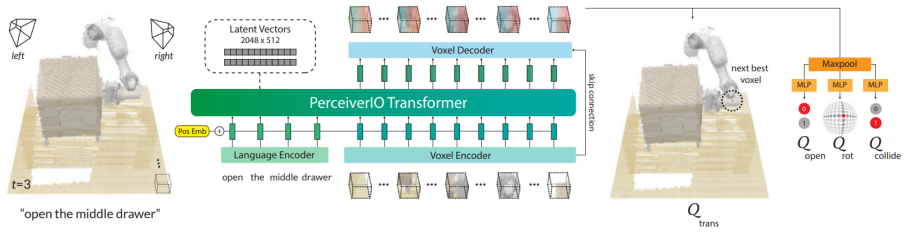
3.2.3. Language conditioned policy for execution

The aforementioned approaches leverage low-level skills for task execution. CLIPorts [86] is a prominent example, utilized as a low-level skill in works that incorporate LLMs for high-level planning [3, 38, 105]. CLIPorts is an end-to-end language-conditioned imitation learning agent that merges the broad semantic understanding provided by CLIP [81] with the spatial precision of Transporter [106], as illustrated in Figure 3.3a. Its semantic stream enhances its ability to comprehend and generalize semantically to some extent, enabling it to tackle open-language tasks in previously unseen scenarios. Like CLIPort, PerAct extends semantic information integration into end-to-end language-conditioned policy learning, progressing from 2D to 3D contexts. It utilizes FiLM [78] for language feature conditioning within a transformer architecture, facilitating the imitation of 6-DoF manipulation tasks from just a few demonstrations. PERACT [87], as depicted in Figure 3.3b proposes the representation of observation and action spaces using 3D voxels and leverages the structured 3D nature of voxel patches to enhance the efficiency of language-conditioned behavioral cloning.

Different from CLIPort [86] or PerAct [87] that use foundation models to achieve open-language policy learning, some works use large-scale data to achieve task generalization. BC-Z [42] is a task-conditioned imitation learning method that has been trained on a large-scale dataset comprising 25,000 episodes across 100 manipulation tasks. It is also employed as a low-level policy in the SayCan framework [3]. BC-Z incorporates a language encoder and a video encoder to train an embedding-conditioned policy based on the encoded embedding z, as illustrated in Figure 3.4c. By encoding input language or the



(a) CLIPort [86]: a two-stream architecture handling spatial and semantic input



(b) PerAct [87]: a transformer framework with language input and voxel state representation

Figure 3.3: SOTA methods for language conditioned policy in 2D and 3D manipulation

current video, BC-Z can effectively generalize to unseen tasks, enhancing its applicability and versatility in real-world scenarios.

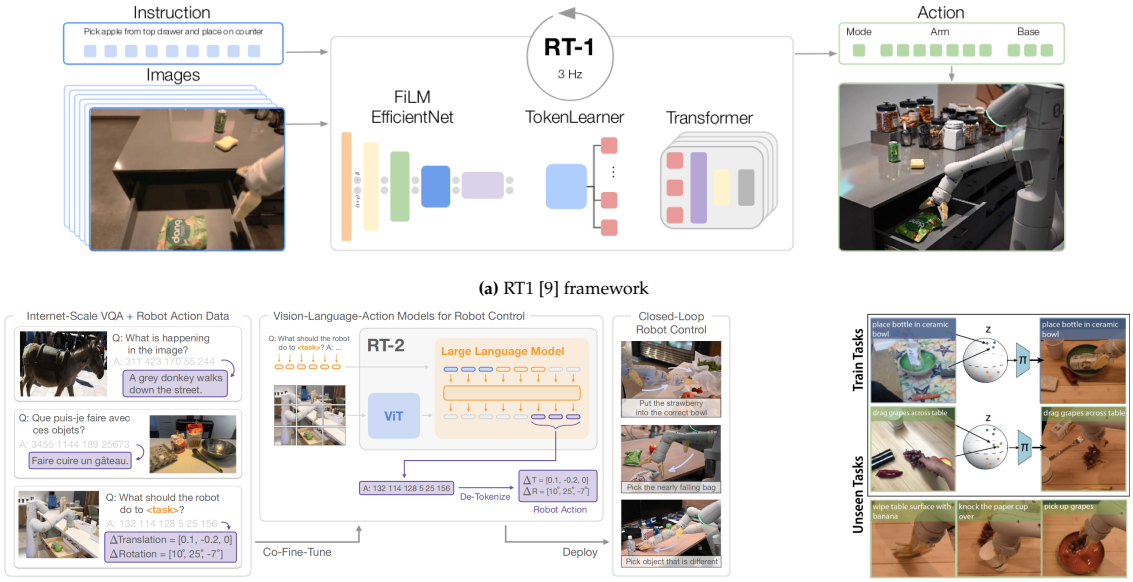
RT-1 [9] aims to leverage large-scale data to develop a general robotic model that processes both language and visual inputs. As depicted in Figure 3.4a, the model is based on a transformer architecture that handles diverse data, including robot trajectories encompassing multiple tasks, objects, and environments. It is trained on a large dataset of real-world robotic experiences, consisting of over 130,000 episodes that cover more than 700 tasks, demonstrating impressive generalization capabilities. Building on the foundations of RT-1 [9], RT-2 [10] extends this approach by training vision-language models on both robotic trajectory data and Internet-scale vision-language tasks. In RT-2, actions are represented as text tokens incorporated directly into the model's training set, treated similarly to natural language tokens. This approach enables RT-2 to harness various capabilities derived from Internet-scale training, including significantly improved generalization to novel objects, the ability to comprehend commands not included in the robot training data, and enhanced reasoning abilities.

Research in RT-X [76] focuses on training a "generalist" robot policy using large-scale data across various tasks and embodiments. The team has compiled a dataset of 1 million episodes featuring 22 different robots across 527 skills. This extensive dataset has been used to train RT-1 [9] and RT-2 [10], demonstrating that these models exhibit positive transfer and significantly enhance the capabilities of multiple robots by leveraging experiences from other platforms.

3.2.4. Can LLM do planning?

Although many studies have already utilized LLMs for planning in reasoning tasks and task-level planning in robotics, some pessimistic views question their capability for genuine reasoning. Some critics refer to LLMs as "Causal Parrots" [104], suggesting that LLMs do not possess the strong reasoning capabilities of humans but rather learn from the distributions of human language data. The term "parrot" implies that LLMs are good at repeating the language found in large datasets. Because they are trained on enormous volumes of contextual data, they can sometimes appear "causal" by mimicking the correlations between causal facts encountered during training.

Further research suggests that LLMs may struggle with generating actionable plans autonomously [45].



(a) RT1 [9] framework

(b) RT2 [10] framework

(c) BC-Z [42]

Figure 3.4: language conditioned robot policy trained on large-scale data

Results from [94] show that only about 12 percent of the plans generated by top-performing LLMs like GPT-4 are executable without errors and achieve their intended goals. These studies also highlight that LLMs cannot self-improve through iterative prompting. Critics argue that unless LLMs are trained not just on "correct data" but also with "corrections data", there is no inherent reason to believe their outputs would be relevant or accurate. Nevertheless, it is suggested that LLMs should be regarded as universal approximate knowledge sources, which have significant potential to contribute to planning and reasoning tasks beyond merely acting as simple front-end/back-end format translators.

From my perspective, LLMs can plan and perform reasoning tasks when utilized appropriately. Using LLMs directly for planning, however, results in infeasible or suboptimal plans. Previously mentioned approaches, such as world grounding, TAMP methods, or low-level skill libraries, enhance the capability of LLMs by bridging the gap between open language and tangible world interactions. LLMs can also serve as universal approximate knowledge sources, providing valuable support in robot learning.

Another concern addressed in this section is whether the combination of groundings, motion planning, and predefined skills is sufficient for solving robotic tasks as an intelligent agent. Robot manipulation tasks often require interactions with various objects, where the complex dynamics involved make it challenging to resolve solely through coding, symbolic planning, or pure motion planning. While pre-trained skill libraries developed through behavior cloning or RL can address contact-rich problems, these low-level skills often lack awareness of high-level plans. This disconnect means that even if a task-level plan appears correct symbolically or linguistically, it may result in execution errors. Consider, for example, a robot commanded to place a chocolate box in a cupboard, with the plan outlined as:

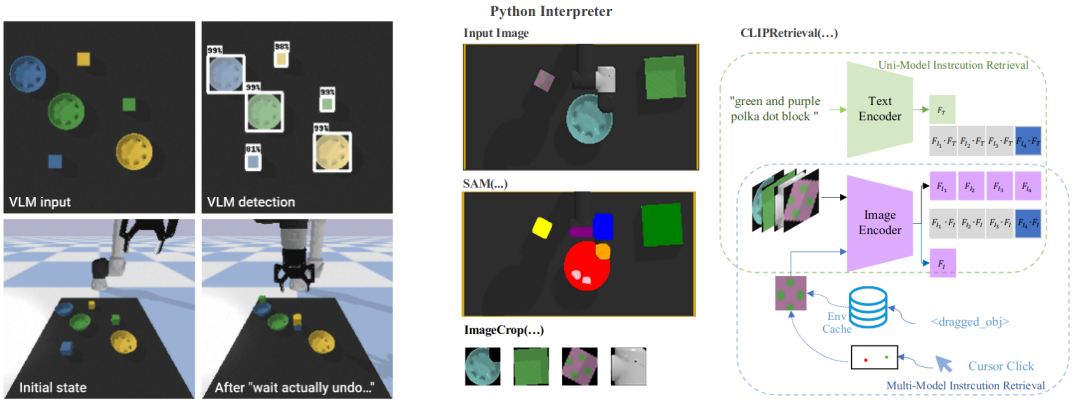
1. pick up the box,
2. move to the cupboard.

During the pre-training phase, the first skill is trained to pick up the box from the top, while the second skill involves holding the box from the side to move it to the cupboard. It is worth noting that holding the box from the top could lead to a collision with the cupboard. Despite the task-level plan being symbolically correct, it fails in execution. To effectively complete such manipulation tasks, there needs to be greater integration and knowledge sharing between the task-level planning and execution levels.

3.3. Foundation Models as Perception

LLMs, with their language modality, can function as the 'brain' of a robotic system. However, like humans, a robot cannot operate effectively without sensory input. Previously, perception modules were typically neural networks trained specifically for recognizing a set of objects or were part of an end-to-end visuomotor policy that directly mapped observations to actions. With the developments in vision foundation models, many recent works have adopted them as perception modules capable of handling open-language tasks in unseen scenarios. This section focuses on VLMs' roles in the literature and explores how they are implemented in these contexts.

Initially, VLMs were utilized primarily as scene descriptors. As depicted in Figure 3.5a, Socratic Models [105] demonstrate how to integrate LLMs with VLMs by using ViLD [26] to describe scene objects, which are then input into an LLM for task-level planning. However, this method uses language as an intermediate from VLM to LLM prompts, which results in the loss of rich visual information and limits the ability to tackle complex tasks. Voxposer [40] proposes using VLMs to construct a global cost map, enhancing performance beyond merely using VLMs as descriptors. It leverages OWL-ViT [64] to generate bounding boxes for objects and SAM [47] to create masks. Instead of using VLMs merely as intermediaries for LLM task-level planning, Voxposer directly utilizes VLM-provided information as a cost map for motion planning, enriching the environmental data available.



(a) VILD [26] for open language detection

(b) SAM [47]+CLIP [81] for detection and segmentation

Figure 3.5: VLMs for perception

In other works [88, 35, 107], VLMs function within the code. LLMs are prompted to generate code functions that actively utilize VLMs for scene information acquisition. In ProgPrompt [88], similar to the Socratic models, ViLD [26] is used to detect scene objects. However, Instruct2Act [35] and DROC [107] utilize a pipeline that employs SAM [47] and CLIP [81] to segment visual inputs into masks and recognize them contrastively, providing a more accurate and flexible perception approach, as depicted in Figure 3.5b.

In DoReMi [27], the visual question answering model Blip [52] is employed as an error detection mechanism to identify any constraints violations in robot motion.

3.4. Foundation Models as Correction/Error Handling

LLMs can function not only as planning and perception modules but also as interactive tools capable of processing feedback due to their inherent language capabilities. By receiving feedback from humans or the environment, LLMs can reason about this feedback and refine their plans accordingly.

Human feedback provides an intuitive method for interacting with robots. Since humans inherently possess strong reasoning abilities, incorporating human feedback can effectively improve planning. DROC [107] introduces a method that distills knowledge from online human feedback to enhance task-level planning. Initially, it generates a task plan using the code-generating capabilities of the LLM, which may be imperfect or suboptimal. After executing the plan, a correction handler assesses and classifies the feedback, determining whether corrections are required at low-level or high-level aspects of the plan. A Knowledge Distillation module then extracts task-relevant knowledge from this feedback,

enabling robots to autonomously generate and refine plans with human input across various tasks. CoPal [44] proposes a system architecture that enables interplay between multiple cognitive levels, including reasoning, planning, and motion generation The system presents a hierarchical architecture for robot manipulation tasks in 3 levels and defines the workflow and feedback between those levels. This architecture defines a multi-level feedback loop that facilitates closed-loop task planning.

Incorporating human feedback is more challenging than handling feedback from dense sensory inputs, as it requires a robust reasoning framework. DoReMi [27] actively uses a VLM to determine whether pre-defined constraints by the LLM are violated. This online detection-feedback-replan loop allows the robot to adapt based on environmental feedback. REFLECT [58] introduces a method for automatically detecting and explaining its failures. It transforms multi-sensory data into a hierarchical summary of the robot's past experiences and queries the LLM with a failure explanation algorithm. This summary detects key changes across three levels: subgoal, event, and sensory input. The LLM is prompted sequentially to identify the error's level, enabling it to correct failures in the task. CLAIRify [89] presents a novel approach that combines automatic iterative prompting to handle syntactic errors and incorporate environmental constraints. LLMs may struggle with generating domain-specific language if not extensively trained on it, leading to errors. CLAIRify receives feedback from a structured verifier that manages environmental constraints. Based on this feedback, the LLM iteratively plans until no further corrections are needed, enhancing the reliability and accuracy of task execution.

3.5. Conclusion

LLMs can play multiple roles in robotics, such as planning, perception, and error handling. Although some critiques label LLMs as "causal parrots" that lack genuine planning capabilities, they can still provide valuable task grounding. LLMs can be effectively applied in robotics when combined with various world grounding techniques. Moreover, given sufficient context, LLMs are capable of in-context learning and making corrections based on identified errors.

However, their practical application in real-world robotics still faces significant challenges, particularly when it comes to tasks that require low-level skills for contact-rich manipulation. These low-level skills often do not seamlessly integrate with high-level planning or may require extensive data for training, as seen in models like RT-1 or RT-2. Consequently, there remains a notable gap between the capabilities of LLMs in theoretical or controlled environments and their effectiveness in real-world applications.

4

Foundation Models in RL Training and Exploration

RL faces challenges such as long training periods, difficult exploration, and issues with reward functions, as discussed in Chapter 2. In robot manipulation tasks, RL methods are traditionally limited to short-horizon scenarios due to the exploration challenges inherent in RL. RL struggles with learning high-level reasoning and low-level control simultaneously for long-horizon tasks.

Foundation models, to some extent, can "plan" for robots based on their commonsense knowledge and language reasoning capabilities. These models can perform task-level planning and act as auxiliary modules in robotic manipulation tasks, as discussed in Chapter 3. Recently, some studies have utilized these capabilities within foundation models to guide and enhance reinforcement learning.

This chapter primarily introduces works that use foundation models to improve reinforcement learning data efficiency by guiding training (Section 4.1), shaping reward signals (Section 4.2) and generating data (Section 4.3).

4.1. Guiding Training and Exploration

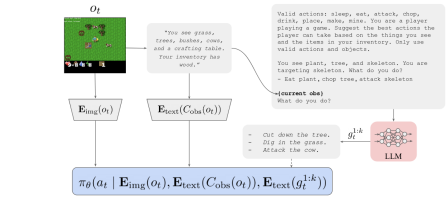
In section 2.3, we introduce the classic method for exploration in RL. Count-based and novelty-motivated methods encourage the RL agent to visit states that are seldom visited or not well-learned. However, in complex environments, such as long-horizon tasks, a novel state does not necessarily represent a meaningful state, as the novelty might be irrelevant to downstream tasks.

ELLM [19], as depicted in Figure 4.1a proposed a method that leverages the knowledge from LLMs to regulate exploration towards human-meaningful behaviors. ELLM allows the LLM to plan a sequence of language goals as intrinsic motivation and measures the similarity between these goals and transitions. The agent is rewarded for achieving higher similarity between the goals and transitions.

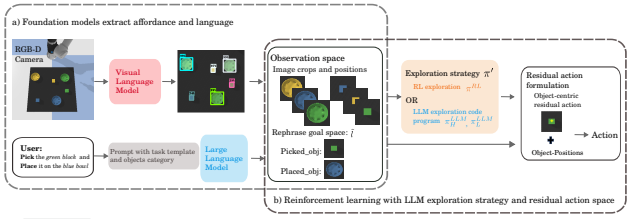
ELLM is the first approach to use LLMs to guide RL exploration towards human-meaningful states. However, there are several limitations to this method. Firstly, the intrinsic motivation reward is based on the similarity to the goals planned by the LLMs, meaning the agent will not learn from other meaningful state distributions outside the LLM plan. Additionally, the rewards derived from language goals can be sparse for low-level continuous control tasks in manipulation.

As illustrated in Figure 4.1b, ExploRLLM [60] also leverages LLM to guide RL agents in taking meaningful actions. Unlike reward shaping, ExploRLLM directly uses LLMs to generate actions that guide the distribution of RL rollouts. Drawing inspiration from the ϵ-greedy strategy, actions generated by LLMs are executed with a probability p, while the RL policy is applied with a probability $1 - p$ during the rollout phase. These transitions are stored in the Replay Buffer, aiding in the faster convergence of off-policy RL methods. Inspired by Code-As-Policy [54], ExploRLLM employs pre-generated code to guide both high-level and low-level actions during exploration. This approach significantly saves time and resources compared to prompting LLMs at every step. However, this method is specifically designed for tabletop manipulation tasks. Taking LLM-guided actions at a fixed frequency does not perform well for guiding exploration in generalized manipulation tasks.

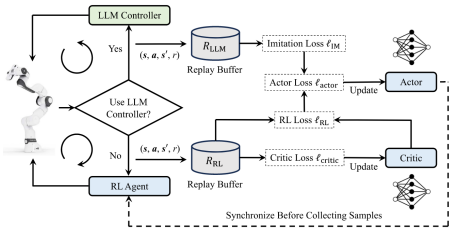
Similarly, RLingua [13], shown in Figure 4.1c, employs an LLM controller to direct RL training for generalized manipulation tasks, using LLM-generated action samples with a decaying probability



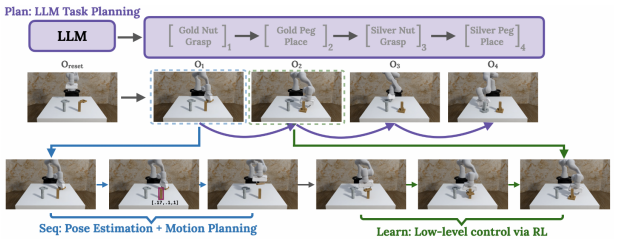
(a) Overview for ELLM [19] methods that use LLM for guiding exploration



(b) ExploRLLM [60]: a framework directly use LLM generate actions for exploration



(c) Rlingua [13]: a framework that utilizes BC loss to guide exploration



(d) PLAN-SEQ-LEARN [16]: a framework guide RL learn contact-rich tasks

Figure 4.1: A illustration of methods using LLMs to guide exploration in RL

during rollouts. Unlike ExploRLLM, RLingua incorporates a behavior cloning loss to train the RL actor, penalizing deviations between the RL-generated actions and those suggested by the LLM. Despite these advancements, RLingua faces challenges in enabling LLMs to accurately generate correct grasp poses, complicating the execution of complex pick-and-place tasks that demand spatial or affordance reasoning. Moreover, the introduction of behavior cloning loss to the training of the policy network may align it too closely with the potentially imperfect policy suggested by the LLM.

Some approaches integrate LLMs and motion planning to guide RL training. Combining these techniques enables RL to avoid complex high-level planning and navigation tasks, concentrating instead on mastering contact-rich manipulation.

One of the representative works is PLAN-SEQ-LEARN [16], illustrated in Figure 4.1d. This modular strategy employs motion planning to bridge the gap between abstract language instructions and learned low-level control, effectively tackling long-horizon robotics tasks from scratch. It utilizes an LLM to generate a high-level plan, orchestrates motion in contact-free regions, and trains a single RL policy to proficiently handle contact-rich tasks. The approach includes a sequence module that moves the robot to regions of interest, thus enabling the RL to learn short-horizon control efficiently.

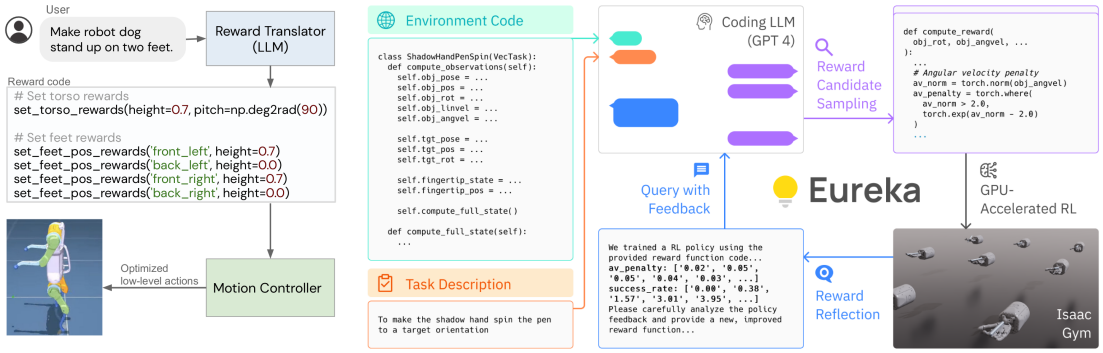
Another development is LEAGUE++ [53], which integrates LLMs with TAMP and RL for continuous skill acquisition. In this framework, the LLM generates a symbolic plan that is subsequently validated through classical motion planning. The RL component then executes these symbolic plans, with the skill training driven by rewards generated by the LLM. This integration demonstrates that a symbolic skills library can significantly enhance training efficiency for tackling new long-horizon manipulation tasks.

4.2. Reward

In the literature concerning applying LLMs to guide RL, reward shaping emerges as a dominant trend. Reward design plays a crucial role in directing policy learning within RL. Additionally, reward design presents challenges in the RL community; it can inadvertently encourage undesired behaviors in robots, and sparse rewards can significantly hinder exploration. To address these challenges, some researchers are exploring using LLMs to generate rewards that aid in reward shaping, thereby encouraging behaviors with semantic meaning.

In [50], the author presents the first framework that utilizes an LLM as a proxy for the reward function. The primary contribution of this work is the design of a textual prompt that includes descriptions and examples of desired behavior, as depicted in Figure 4.2a During training, the LLM assesses the RL's behavior against the desired behavior described in the prompt and generates a corresponding reward signal.

In robotics, authors in [103] employ LLMs to define parameters for dense reward. To bridge the gap between high-level instructions and low-level robotic actions, their approach incorporates a motion descriptor that plans the motion and a reward coder that translates this motion into a code function and parameters. This reward mechanism has been successfully implemented in quadruped and manipulator robots to execute instructions. Similarly, Text2Reward [102] generates dense reward functions based on goals expressed in natural language. The LLM receives a code-style environment description and human feedback and outputs the reward as a Python function. Given the sensitivity of RL training, single-turn generation sometimes fails to produce adequate reward functions to complete tasks. Text2Reward addresses this by soliciting human feedback on failure modes and refining the dense reward accordingly.



(a) Architecture for generating reward for robot skill given open language [103]

(b) Eureka [61]: a framework for LLMs to generate rewards via evolutionary optimization

Figure 4.2: A illustration of methods using LLMs to generate rewards for RL

Previous methods primarily utilized LLMs to generate rewards for relatively simple tasks. However, a more sophisticated approach to prompting LLMs is required for more complex tasks where the reward function is less apparent. Eureka [61], as illustrated in Figure 4.2b, employs evolutionary search to generate multiple executable reward functions. These functions are then evaluated during training. Starting with a reward function from an earlier iteration, EUREKA performs in-context reward mutation, generating an improved reward function based on textual feedback. Without relying on task-specific prompts or predefined reward templates, EUREKA successfully produces reward functions that surpass those engineered by human experts.

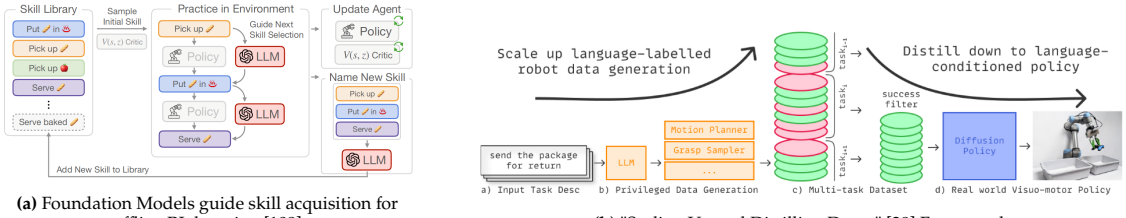
There are also works that utilized VLMs to generate reward signals. VLM-RMs [83] using pretrained VLM CLIP [81] to generate reward signals to train a humanoid robot to learn complex tasks without a manually specified reward function. Those rewards are the cosine similarity between state image representation and natural language task description. The direct result from those VLMs could be noisy or inconsistent. Addressing this, RL-VLM-F [100] introduces a method to learn a reward model based on feedback from VLMs. It queries VLMs to determine preferences over pairs of the agent's image observations, guided by textual descriptions of the task goals and subsequently learns a reward function from these preferences. The authors demonstrate that RL-VLM-F successfully generates rewards for various tasks, including manipulating rigid, articulated, and deformable objects.

4.3. Data/Demonstration Generation

Besides reward shaping, LLMs can also directly influence data collection to guide policy development. Robots can generate rollouts or meaningful actions driven by LLM guidance. This data is then used for skill acquisition in robotics. This method is not confined to online reinforcement learning but is also applicable to offline reinforcement learning and imitation learning.

"Scaling Up and Distilling Down" [29], shown in Figure 4.3b serves as an illustrative example of these methods, although it does not employ RL for learning skills. For scaling up data generation, it uses an LLM to guide high-level planning and sampling-based robot planners to generate diverse manipulation trajectories. Successful trajectories are compiled into a dataset for multitasking and subsequently distilled into a policy for real-world deployment.

In contrast to merely generating datasets, some works focus on generating policies. Bootstrap Your Own Skills (BOSS) [108], as illustrated in Figure 4.3a, is a method that learns to execute a set of useful,



(a) Foundation Models guide skill acquisition for offline RL learning [108]

(b) "Scaling Up and Distilling Down" [29] Framework

Figure 4.3: Foundation models for offline data generation

long-horizon skills with minimal supervision through LLM-guided skill bootstrapping. The LLM plays a crucial role in connecting an initial skill library to create new, extended behavioral sequences. These rollouts are collected as experiences, serving as the training data for an offline RL agent. Newly discovered skill chains, summarized by the LLM, are integrated back into the skill library for further development and refinement. Over time, this iterative process expands the agent's skill repertoire significantly. In this approach, LLMs utilize their comprehensive knowledge to effectively bridge and enhance the skill library, continuously shaping and enriching it with new skills and data.

4.4. Conclusion

This chapter discusses various approaches to integrating reinforcement learning (RL) training with foundation models to enhance the efficiency of RL exploration. As depicted in Figure 4.4, these methods leverage foundation models in distinct ways. One approach involves using the LLM to explicitly plan and guide RL training, which we call "Foundation models guiding training." The "Data collection" strategy employs plans derived from foundation models to direct robot actions. The data generated from these actions then serves as an offline dataset for policy shaping.

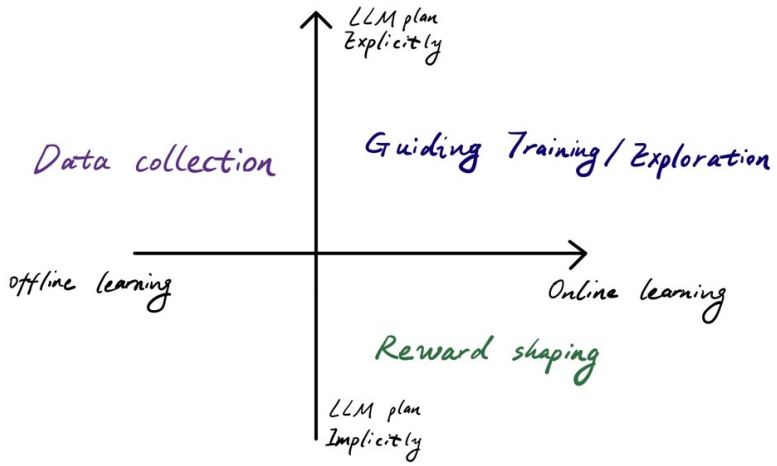


Figure 4.4: Different methods to integrate RL training with Foundation Models

Although these methods show promising results in bridging the reasoning capabilities of foundation models with RL control policy learning, they are not without limitations. A common assumption in most of these approaches is that insights derived from LLMs are accurate, which is expected to lead to improvements in RL. However, suboptimal guidance from LLM knowledge can result in inefficient training or unpredictable behaviors in RL. Particularly with methods like reward shaping, while the rewards generated by LLMs might be contextually relevant, they can still lead to typical RL issues where the reward structure inadvertently encourages undesired behaviors. Eureka [61] attempts to address this problem through evolutionary optimization, which allows for the gradual improvement of policy performance. Nevertheless, this evolutionary process can be extremely resource-intensive and time-consuming, involving numerous training trials to achieve optimal results.

5

Discussion and Conclusion

5.1. Research Question Revisited

This chapter provides an overview of the findings from the literature review. After discussing recent trends and significant developments in the field over the past few years, we now return to the research questions posed in Chapter 1:

1. What are the state-of-the-art (SOTA) reinforcement learning methods for efficient exploration that effectively achieve the exploration-exploitation trade-off? How could incorporating human prior knowledge into RL accelerate the exploration process?
2. How can foundation models contribute to enhancing robotic manipulation? What roles can these models play in robot manipulation, and what are their limitations in practical applications?
3. What recent advancements have been made in integrating foundation models with reinforcement learning?

5.1.1. RL and exploration

The content in Chapter 2 addresses the first research question regarding state-of-the-art reinforcement learning methods for efficient exploration that effectively manage the exploration-exploitation trade-off. Basic RL algorithms utilize stochastic policies, action noise, or entropy regulation to achieve this balance, with SAC [30] providing a foundational approach for many subsequent studies. While successful in various control policies, these basic RL algorithms struggle to tackle long-horizon manipulation tasks independently.

Exploration algorithms explicitly designed for specific applications, such as count-based [4, 6, 75, 62, 92] or novelty-driven [90, 77, 12, 109] methods, show promising performance in navigating mazes or game spaces. However, these methods are not inherently suited for robot manipulation, where most state-action pairs in joint-position or end-effector position joint spaces are less meaningful. Two methodological families have demonstrated high data efficiency and effectiveness in addressing these challenges:

1. **demonstration as prior knowledge**: Using demonstrations as prior knowledge from humans to RL agents allows these agents to bypass initial learning stages by incorporating this data directly into the replay buffer. Techniques such as behavior cloning loss [32, 82, 70, 69], high Update-to-Data ratios [5, 14], ensemble critics [5, 14], and layer normalization [5] are integrated to enhance training efficiency.
2. **Hierarchy modelling:** Hierarchical modeling aligns with the human cognitive approach to structuring robot learning for long-horizon manipulation tasks, enabling robots to understand both what to do (high-level) and how to do it (low-level). This structured approach has proven to facilitate a more efficient exploration process than that achieved by single-layer RL agents, by allowing simultaneous exploration at multiple levels.

These two approaches incorporate prior knowledge into reinforcement learning at different levels, enriching the RL agents' capability to handle complex tasks. Furthermore, there are methods that synergize these approaches [28, 79], combining hierarchical modeling with learned skill priors [79] to further boost training efficiency and effectiveness.

5.1.2. Foundation Models for Manipulation

The content in Chapter 3 addresses the second research question regarding the roles that foundation models can play in robotic manipulation. Recent literature demonstrates that foundation models can function effectively as planning modules [3, 37, 54, 56, 57], perception modules [105, 40], and mechanisms for handling environmental and human feedback [107, 44, 27, 58].

Despite some skepticism regarding their reasoning capabilities [104], these works illustrate that LLMs can facilitate task-level planning in robotic tasks when properly utilized. Effective utilization includes providing world grounding, using well-designed prompts, or TAMP systems.

However, there are still limitations to consider:

1. **Sub-Optimality of Plans:** Plans generated by LLMs may be sub-optimal and may require further refinement to meet operational standards.
2. **Limitations in Low-Level Control:** While LLMs show promising performance in task-level planning, they face challenges in low-level control policy, especially for contact-rich tasks. These low-level skills often do not seamlessly integrate with high-level planning or may require extensive data for training, as seen in models like RT-1 [9] or RT-2 [10].

There remains a noticeable gap between high-level planning and the execution capabilities of low-level policies in manipulation tasks, indicating a need for better integration and alignment of these two layers to enhance overall task performance.

5.1.3. Foundation Models Guiding RL Training and Exploration

The content in Chapter 4 addresses the third research question concerning the integration of foundation models with RL. This integration has led to significant advancements in RL policy learning for robotics, categorized into three main methods:

1. **Foundation models guide training and exploration**,
2. **Foundation models generate reward**,
3. **Foundation models generate offline data**.

These methods have been instrumental in enhancing the training efficiency of reinforcement learning. However, they rely heavily on the assumption that the plans or reward signals generated by the foundation models are sufficiently accurate to guide the RL training process effectively. Particularly, the reward shaping method may encounter challenges if it learns unwanted behaviors or operates inefficiently due to suboptimal or incorrect insights from LLMs.

5.1.4. Limitations

This review has certain limitations that should be considered when interpreting its findings. Firstly, the nature of Large Language Models (LLMs) has been primarily viewed as a tool, without exploring the potential relationship where RL could be used to fine-tune LLMs. This one-sided perspective may overlook important dynamics in the interaction between these technologies.

Secondly, the theoretical background of LLMs has not been extensively covered in this review. Although understanding the theoretical underpinnings of LLMs is crucial for a deeper comprehension of their operations and limitations, it was not the focus of this study, which primarily concentrated on the integration of LLMs in robotic tasks.

Lastly, due to the rapidly evolving nature of this research area, some claims in this literature review may be perceived as subjective and not fully substantiated. The field's quick advancement means that even the most recent studies can quickly become outdated, necessitating continuous updates and revisions to stay current with new developments and insights.

5.2. Future work and Discussion

Although current literature aiming to integrate reinforcement learning with foundation models has shown promising performance, most studies still rely on the assumption that the outputs provided by these models, such as plans, are correct. RL learning could also benefit from providing foundation models with environmental feedback, which would help in self-correction. Additionally, some methods from classic reinforcement learning literature that aim to improve training efficiency have not been

widely combined with foundation models, such as Hierarchical RL or RL with demonstrations. In Figure 5.1, we illustrate the plan that integrates RL with foundation models to utilize the advantages from both sides.

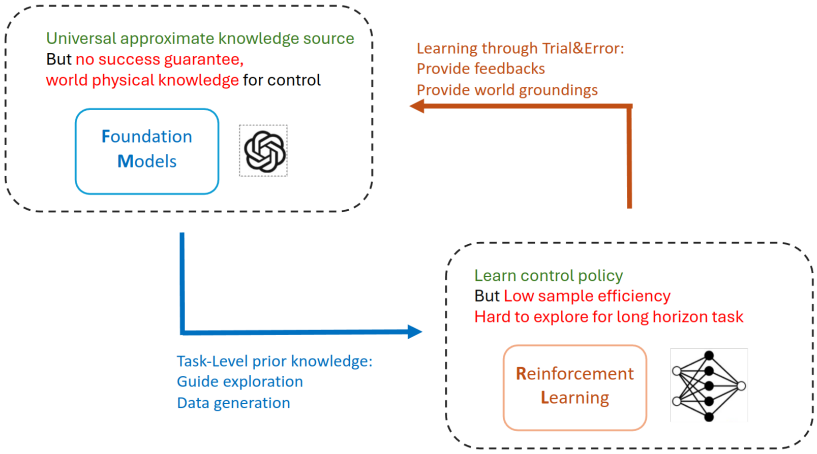


Figure 5.1: Integrating RL with Foundation models

To address existing challenges within both the reinforcement learning and foundation models literature, we propose several directions for future work:

1. **LLM self-improvement through RL** If the plans or rewards generated from foundation models are suboptimal, this can limit the efficiency of RL training. While Eureka [61] utilizes evolutionary optimization to enhance insights from foundation models, training thousands of episodes is resource-intensive. However, during training, there is potential to refine the plans generated by LLMs using episode feedback from RL. Previous methods have demonstrated that LLMs can reflect on their own failures based on environmental feedback. This approach would allow LLMs' plan and low-level control policy to co-evolve through trial and error within an RL framework, potentially leading to more effective learning strategies.
2. **LLM is compatible in hierarchy RL framework** In hierarchical reinforcement learning HRL, action in the first layer often encapsulates rich semantic meaning, outlining subgoals for manipulation tasks. This characteristic makes LLMs highly compatible with task-level planning within HRL frameworks. LLMs can guide the high-level layer of HRL, enabling efficient exploration at this level and facilitating a more structured approach to achieving complex objectives.
3. **LLM data generation with Efficient RL** LLMs have demonstrated their capability to generate robotic trajectories, as shown in [29]. These trajectories can serve as prior data in efficient RL frameworks like those discussed in [5], potentially enhancing the efficiency of RL training. This approach addresses the domain shift issues common in previous methods primarily relying on behavior cloning policies. Using online RL to fine-tune these LLM-generated trajectories, it is able to bridge the gap between simulated training environments and real-world applications, ensuring that the learned behaviors are effective and adaptable.

5.3. Conclusion

This literature review examines the current research on utilizing foundation models to guide reinforcement learning, with a focus on RL theory and exploration methods outlined in Chapter 2, the functionality of foundation models discussed in Chapter 3, and the integration of these two fields presented in Chapter 4. Our goal is to enhance the efficiency of reinforcement learning by leveraging the knowledge encapsulated in foundation models. We have identified that there remains a gap between current studies and enabling LLM robust generate guidance improving RL efficiency for long-horizon manipulation tasks.

As for the next stage, we aim to develop a method that synergistically integrates foundation models with reinforcement learning. Our goal is to design a framework that enables foundation models to guide RL learning in a more accurate and efficient manner, thereby bridging the identified gaps and enhancing overall system performance.

References

[1] Josh Achiam et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] Joshua Achiam. "Openai spinning up". In: *GitHub, GitHub repository* (2018).

[3] Michael Ahn et al. "Do as i can, not as i say: Grounding language in robotic affordances". In: *arXiv preprint arXiv:2204.01691* (2022).

[4] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. "Exploration–exploitation tradeoff using variance estimates in multi-armed bandits". In: *Theoretical Computer Science* 410.19 (2009), pp. 1876–1902.

[5] Philip J Ball et al. "Efficient online reinforcement learning with offline data". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 1577–1594.

[6] Marc Bellemare et al. "Unifying count-based exploration and intrinsic motivation". In: *Advances in neural information processing systems* 29 (2016).

[7] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. Vol. 4. Athena scientific, 2012.

[8] Rishi Bommasani et al. "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258* (2021).

[9] Anthony Brohan et al. "Rt-1: Robotics transformer for real-world control at scale". In: *arXiv preprint arXiv:2212.06817* (2022).

[10] Anthony Brohan et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *arXiv preprint arXiv:2307.15818* (2023).

[11] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[12] Yuri Burda et al. "Exploration by random network distillation". In: *arXiv preprint arXiv:1810.12894* (2018).

[13] Liangliang Chen et al. "RLingua: Improving Reinforcement Learning Sample Efficiency in Robotic Manipulations With Large Language Models". In: *IEEE Robotics and Automation Letters* (2024).

[14] Xinyue Chen et al. "Randomized ensembled double q-learning: Learning fast without a model". In: *arXiv preprint arXiv:2101.05982* (2021).

[15] Yongchao Chen et al. "Autotamp: Autoregressive task and motion planning with llms as translators and checkers". In: *arXiv preprint arXiv:2306.06531* (2023).

[16] Murtaza Dalal et al. "Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks". In: *arXiv preprint arXiv:2405.01534* (2024).

[17] Norman Di Palo et al. "Towards a unified agent with foundation models". In: *arXiv preprint arXiv:2307.09668* (2023).

[18] Yan Ding et al. "Task and motion planning with large language models for object rearrangement". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 2086–2092.

[19] Yuqing Du et al. "Guiding pretraining in reinforcement learning with large language models". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 8657–8677.

[20] Íñigo Elguea-Aguinaco et al. "A review on reinforcement learning for contact-rich robotic manipulation tasks". In: *Robotics and Computer-Integrated Manufacturing* 81 (2023), p. 102517.

[21] Meng Fang et al. "Curriculum-guided hindsight experience replay". In: *Advances in neural information processing systems* 32 (2019).

[22] Carlos Florensa et al. “Automatic goal generation for reinforcement learning agents”. In: *International conference on machine learning*. PMLR. 2018, pp. 1515–1528.

[23] Carlos Florensa et al. “Reverse curriculum generation for reinforcement learning”. In: *Conference on robot learning*. PMLR. 2017, pp. 482–495.

[24] Maria Fox and Derek Long. “PDDL2. 1: An extension to PDDL for expressing temporal planning domains”. In: *Journal of artificial intelligence research* 20 (2003), pp. 61–124.

[25] Scott Fujimoto, Herke Hoof, and David Meger. “Addressing function approximation error in actor-critic methods”. In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.

[26] Xiuye Gu et al. “Open-vocabulary object detection via vision and language knowledge distillation”. In: *arXiv preprint arXiv:2104.13921* (2021).

[27] Yanjiang Guo et al. “Doremi: Grounding language model by detecting and recovering from plan-execution misalignment”. In: *arXiv preprint arXiv:2307.00329* (2023).

[28] Abhishek Gupta et al. “Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning”. In: *arXiv preprint arXiv:1910.11956* (2019).

[29] Huy Ha, Pete Florence, and Shuran Song. “Scaling up and distilling down: Language-guided robot skill acquisition”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 3766–3777.

[30] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[31] Matteo Hessel et al. “Rainbow: Combining improvements in deep reinforcement learning”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[32] Todd Hester et al. “Deep q-learning from demonstrations”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[33] Yafei Hu et al. “Toward general-purpose robots via foundation models: A survey and meta-analysis”. In: *arXiv preprint arXiv:2312.08782* (2023).

[34] Yingdong Hu et al. “Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning”. In: *arXiv preprint arXiv:2311.17842* (2023).

[35] Siyuan Huang et al. “Instruct2act: Mapping multi-modality instructions to robotic actions with large language model”. In: *arXiv preprint arXiv:2305.11176* (2023).

[36] Wei Huang et al. “How Good Are Low-bit Quantized LLaMA3 Models? An Empirical Study”. In: *arXiv preprint arXiv:2404.14047* (2024).

[37] Wenlong Huang et al. “Grounded decoding: Guiding text generation with grounded models for robot control”. In: *arXiv preprint arXiv:2303.00855* (2023).

[38] Wenlong Huang et al. “Inner monologue: Embodied reasoning through planning with language models”. In: *arXiv preprint arXiv:2207.05608* (2022).

[39] Wenlong Huang et al. “Language models as zero-shot planners: Extracting actionable knowledge for embodied agents”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 9118–9147.

[40] Wenlong Huang et al. “Voxposer: Composable 3d value maps for robotic manipulation with language models”. In: *arXiv preprint arXiv:2307.05973* (2023).

[41] Boris Ivanovic et al. “Barc: Backward reachability curriculum for robotic reinforcement learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 15–21.

[42] Eric Jang et al. “Bc-z: Zero-shot task generalization with robotic imitation learning”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 991–1002.

[43] Albert Q Jiang et al. “Mistral 7B”. In: *arXiv preprint arXiv:2310.06825* (2023).

[44] Frank Joublin et al. “Copal: Corrective planning of robot actions with large language models”. In: *arXiv preprint arXiv:2310.07263* (2023).

[45] Subbarao Kambhampati et al. “LLMs Can’t Plan, But Can Help Planning in LLM-Modulo Frameworks”. In: *arXiv preprint arXiv:2402.01817* (2024).

[46] Kento Kawaharazuka et al. “Real-World Robot Applications of Foundation Models: A Review”. In: *arXiv preprint arXiv:2402.05741* (2024).

[47] Alexander Kirillov et al. “Segment anything”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026.

[48] Jens Kober, J Andrew Bagnell, and Jan Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[49] Tejas D Kulkarni et al. “Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation”. In: *Advances in neural information processing systems* 29 (2016).

[50] Minae Kwon et al. “Reward design with language models”. In: *arXiv preprint arXiv:2303.00001* (2023).

[51] Seunghyun Lee et al. “Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 1702–1712.

[52] Junnan Li et al. “Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models”. In: *International conference on machine learning*. PMLR. 2023, pp. 19730–19742.

[53] Zhaoyi Li et al. “EMPOWERING CONTINUAL ROBOT LEARNING THROUG GUIDED SKILL ACQUISITION WITH LANGUAGE MODELS”. In: *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

[54] Jacky Liang et al. “Code as policies: Language model programs for embodied control”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 9493–9500.

[55] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[56] Kevin Lin et al. “Text2motion: From natural language instructions to feasible plans”. In: *Autonomous Robots* 47.8 (2023), pp. 1345–1365.

[57] Bo Liu et al. “Llm+ p: Empowering large language models with optimal planning proficiency”. In: *arXiv preprint arXiv:2304.11477* (2023).

[58] Zeyi Liu, Arpit Bahety, and Shuran Song. “Reflect: Summarizing robot experiences for failure explanation and correction”. In: *arXiv preprint arXiv:2306.15724* (2023).

[59] Sha Luo, Hamidreza Kasaei, and Lambert Schomaker. “Accelerating reinforcement learning for reaching using continuous curriculum learning”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[60] Runyu Ma et al. “ExploRLLM: Guiding Exploration in Reinforcement Learning with Large Language Models”. In: *arXiv preprint arXiv:2403.09583* (2024).

[61] Yecheng Jason Ma et al. “Eureka: Human-level reward design via coding large language models”. In: *arXiv preprint arXiv:2310.12931* (2023).

[62] Jarryd Martin et al. “Count-based exploration in feature space for reinforcement learning”. In: *arXiv preprint arXiv:1706.08090* (2017).

[63] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. “Scaling open-vocabulary object detection”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[64] Matthias Minderer et al. “Simple open-vocabulary object detection”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 728–755.

[65] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[66] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[67] Ofir Nachum et al. “Data-efficient hierarchical reinforcement learning”. In: *Advances in neural information processing systems* 31 (2018).

[68] Ofir Nachum et al. “Why does hierarchy (sometimes) work so well in reinforcement learning?” In: *arXiv preprint arXiv:1909.10618* (2019).

[69] Ashvin Nair et al. "Awac: Accelerating online reinforcement learning with offline datasets". In: *arXiv preprint arXiv:2006.09359* (2020).

[70] Ashvin Nair et al. "Overcoming exploration in reinforcement learning with demonstrations". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 6292–6299.

[71] Mitsuhiko Nakamoto et al. "Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning". In: *Advances in Neural Information Processing Systems* 36 (2024).

[72] Soroush Nasiriany, Huihan Liu, and Yuke Zhu. "Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 7477–7484.

[73] R OpenAI. "Gpt-4 technical report. arxiv 2303.08774". In: *View in Article* 2.5 (2023).

[74] Maxime Oquab et al. "Dinov2: Learning robust visual features without supervision". In: *arXiv preprint arXiv:2304.07193* (2023).

[75] Georg Ostrovski et al. "Count-based exploration with neural density models". In: *International conference on machine learning*. PMLR. 2017, pp. 2721–2730.

[76] Abhishek Padalkar et al. "Open x-embodiment: Robotic learning datasets and rt-x models". In: *arXiv preprint arXiv:2310.08864* (2023).

[77] Deepak Pathak et al. "Curiosity-driven exploration by self-supervised prediction". In: *International conference on machine learning*. PMLR. 2017, pp. 2778–2787.

[78] Ethan Perez et al. "Film: Visual reasoning with a general conditioning layer". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[79] Karl Pertsch, Youngwoon Lee, and Joseph Lim. "Accelerating reinforcement learning with learned skill priors". In: *Conference on robot learning*. PMLR. 2021, pp. 188–204.

[80] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[81] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[82] Aravind Rajeswaran et al. "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations". In: *arXiv preprint arXiv:1709.10087* (2017).

[83] Juan Rocamonde et al. "Vision-language models are zero-shot reward models for reinforcement learning". In: *arXiv preprint arXiv:2310.12921* (2023).

[84] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[85] John Schulman et al. "Trust region policy optimization". In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[86] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Cliport: What and where pathways for robotic manipulation". In: *Conference on robot learning*. PMLR. 2022, pp. 894–906.

[87] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Perceiver-actor: A multi-task transformer for robotic manipulation". In: *Conference on Robot Learning*. PMLR. 2023, pp. 785–799.

[88] Ishika Singh et al. "Progprompt: Generating situated robot task plans using large language models". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 11523–11530.

[89] Marta Skreta et al. "Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting". In: *arXiv preprint arXiv:2303.14100* (2023).

[90] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. "Incentivizing exploration in reinforcement learning with deep predictive models". In: *arXiv preprint arXiv:1507.00814* (2015).

[91] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[92] Haoran Tang et al. "# exploration: A study of count-based exploration for deep reinforcement learning". In: *Advances in neural information processing systems* 30 (2017).

[93] Hugo Touvron et al. “Llama 2: Open foundation and fine-tuned chat models”. In: *arXiv preprint arXiv:2307.09288* (2023).

[94] Karthik Valmeekam et al. “On the planning abilities of large language models-a critical investigation”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 75993–76005.

[95] Hado Van Hasselt, Arthur Guez, and David Silver. “Deep reinforcement learning with double q-learning”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[96] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[97] Mel Vecerik et al. “Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards”. In: *arXiv preprint arXiv:1707.08817* (2017).

[98] Alexander Sasha Vezhnevets et al. “Feudal networks for hierarchical reinforcement learning”. In: *International conference on machine learning*. PMLR. 2017, pp. 3540–3549.

[99] Shu Wang et al. “LLMˆ 3: Large Language Model-based Task and Motion Planning with Motion Failure Reasoning”. In: *arXiv preprint arXiv:2403.11552* (2024).

[100] Yufei Wang et al. “RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback”. In: *arXiv preprint arXiv:2402.03681* (2024).

[101] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[102] Tianbao Xie et al. “Text2reward: Automated dense reward function generation for reinforcement learning”. In: *arXiv preprint arXiv:2309.11489* (2023).

[103] Wenhao Yu et al. “Language to rewards for robotic skill synthesis”. In: *arXiv preprint arXiv:2306.08647* (2023).

[104] Matej Zečević et al. “Causal parrots: Large language models may talk causality but are not causal”. In: *arXiv preprint arXiv:2308.13067* (2023).

[105] Andy Zeng et al. “Socratic models: Composing zero-shot multimodal reasoning with language”. In: *arXiv preprint arXiv:2204.00598* (2022).

[106] Andy Zeng et al. “Transporter networks: Rearranging the visual world for robotic manipulation”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 726–747.

[107] Lihan Zha et al. “Distilling and retrieving generalizable knowledge for robot manipulation via language corrections”. In: *arXiv preprint arXiv:2311.10678* (2023).

[108] Jesse Zhang et al. “Bootstrap your own skills: Learning to solve new tasks with large language model guidance”. In: *arXiv preprint arXiv:2310.10021* (2023).

[109] Tianjun Zhang et al. “Noveld: A simple yet effective exploration criterion”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 25217–25230.

[110] Xiaohan Zhang et al. “Visually grounded task and motion planning for mobile manipulation”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 1925–1931.