2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# LLM-Guided Task- and Affordance-Level Exploration in Reinforcement Learning

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

*Abstract*— **Reinforcement learning (RL) is a promising approach for robotic manipulation, but it can suffer from low sample efficiency and requires extensive exploration of large state-action spaces. Recent methods leverage the commonsense knowledge and reasoning abilities of large language models (LLMs) to guide exploration toward more meaningful states. However, LLMs can produce plans that are semantically plausible yet physically infeasible, yielding unreliable behavior. We introduce LLM-TALE, a framework that uses LLMs' planning to directly steer RL exploration. LLM-TALE integrates planning at both the task level and the affordance level, improving learning efficiency by directing agents toward semantically meaningful actions. Unlike prior approaches that assume optimal LLM-generated plans or rewards, LLM-TALE corrects suboptimality online and explores multimodal affordance-level plans without human supervision. We evaluate LLM-TALE on pick-and-place tasks in standard RL benchmarks, observing improvements in both sample efficiency and success rates over strong baselines. Real-robot experiments indicate promising zero-shot sim-to-real transfer. Code and supplementary material are available at `https://llm-tale.github.io`.**

## I. Introduction

Reinforcement learning (RL) [1] offers a powerful framework for learning decision-making and control policies in robotics [2] through interaction with the environment. However, practical deployment is hindered by low sample efficiency. Training stable manipulation policies requires exhaustive exploration of vast state–action spaces and adequate reward feedback. This challenge is especially problematic when policies are randomly initialized in long-horizon tasks with sparse rewards. To overcome this problem, intrinsic motivation methods have been introduced to encourage visiting novel states via intrinsic rewards [3]–[7], but the resulting novelty signals can misalign with task-relevant behavior. A well-known failure mode for some intrinsic-motivation methods is the "noisy-TV" thought experiment: an agent can be drawn to a TV showing white noise, since each frame is novel and effectively unpredictable.

Another line of work improves efficiency by incorporating demonstrations that inject human knowledge into off-policy RL [8]–[13], but this relies on costly human demonstrations.

Foundation models such as Llama 3 [14] and GPT-4 [15], trained on vast datasets, offer a scalable alternative. These models act as approximate knowledge sources [16], leveraging human-like reasoning to aid robotic manipulation. Recent work shows that large language models (LLMs) and vision–language models (VLMs) can interpret environmental context and perform task-level reasoning, translating open-language commands into sequences of executable skills [17]–[21]. However, their understanding of the physical world is incomplete and can yield erroneous guidance. Despite these errors, their approximation of human knowledge can enhance RL training by guiding it at higher levels of abstraction (e.g., task or affordance levels). Using LLMs to generate dense reward functions [22]–[25] aligns robot behavior with human language, mitigating challenges posed by sparse rewards. Yet such human-like rewards can induce undesirable behavior, e.g., remaining in high-reward regions without accomplishing the task. Beyond rewards, recent work [26], [27] uses LLMs to generate actions directly, guiding robots toward semantically meaningful regions and shifting the data distribution in off-policy replay buffers. These exploration-driven actions may be suboptimal initially but improve during policy learning. Nonetheless, the effectiveness of these methods depends heavily on the quality of LLM-generated actions and, at present, is largely limited to off-policy RL.

In this paper, we introduce *LLM-guided Task- and Affordance-Level Exploration* (LLM-TALE), a method that steers exploration toward semantically meaningful regions of the state space by leveraging LLM guidance that is more reliable at higher levels of abstraction (i.e., task and affordance). Because affordance-level actions are often multimodal and some modes are semantically valid yet physically infeasible, our method uses the critic's value estimates (e.g., Q-values) to explore more promising modes and to avoid fruitless exploration of infeasible actions.

This work introduces LLM-TALE and makes the following contributions:

1) We propose a hierarchical, LLM-driven planning scheme that generates task-level plans and affordance-level action candidates.
2) We present a goal-conditioned residual RL framework in which goals are derived from LLM-generated affordances, and exploration is guided by intrinsic rewards defined relative to these goals.
3) We introduce critic- and uncertainty-guided affordance-level exploration over LLM-generated proposals, enabling a trade-off between exploration and exploitation across affordance modalities.

These components bias exploration toward semantically meaningful regions of the action space. Our experiments show high sample efficiency in sparse-reward robotic manipulation for both on- and off-policy RL, while real-world evaluations show promising zero-shot sim-to-real transfer.

* Equal Contribution. [1] Cognitive Robotics, Delft University of Technology, The Netherlands. [2] RWTH Aachen University, Germany.

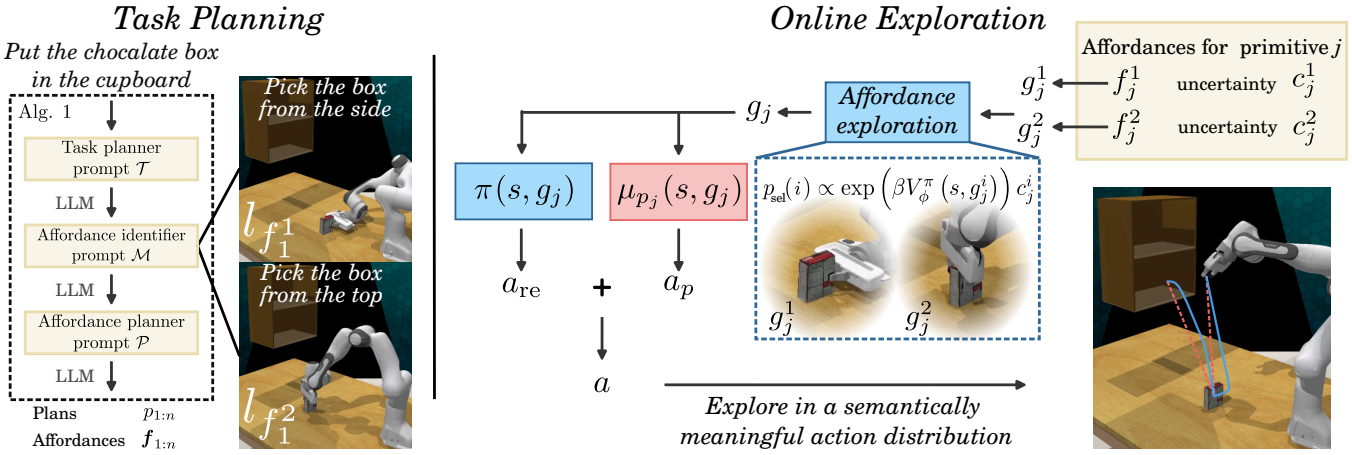



Fig. 1: LLM-guided Task- and Affordance-Level Exploration (LLM-TALE) uses LLMs to generate task- and affordance-level plans to explore semantically meaningful regions. It explores multimodal affordances using goal-conditioned value functions.

## II. Related Work

### A. Exploration in Reinforcement Learning

Classic exploration methods in reinforcement learning provide intrinsic rewards that encourage visiting novel or uncertain states [3]–[6]. While effective at avoiding exploitative actions, such signals can accumulate low-value experience in robotic manipulation. To address sparse rewards, prior works integrate human demonstrations into replay buffers and guide learning with behavior cloning (BC) losses [8], [9], [11]. RLPD [10] achieves strong data efficiency by using high update-to-data ratios and ensemble critics to learn from offline data without BC. IBRL [12] initializes TD3 agents [28] from demonstration-trained policies. However, these approaches require high-quality demonstrations.

### B. Reinforcement Learning with Foundation Models

Foundation models enable task-level reasoning for robotics [18], [29]–[33]. In RL, they have been used to specify dense reward functions aligned with human language [22]–[24], though such rewards can lead to unintended behaviors such as reward hacking (e.g., action repetition without task completion). Eureka [25] refines reward code via evolutionary optimization in parallel environments, but the evolutionary loop can be computationally costly. Recent work [26], [27] instead uses foundation models to propose actions directly, guiding robots toward semantically meaningful regions and shifting the data distribution in off-policy replay buffers. However, such approaches remain dependent on high-quality LLM actions for success.

### C. Research Gap

In contrast to existing methods, LLM-TALE learns an RL policy that operates in a residual action space under LLM guidance to focus exploration on semantically meaningful regions. By doing so, LLM-TALE induces a more informative online data distribution, without the need for high-quality (human) demonstrations. Moreover, LLM-TALE addresses LLM reasoning errors online through interaction and explores multimodal affordances identified by the task planner.

## III. Preliminaries

### A. Reinforcement Learning

We model the environment 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 $R$ provides reward $r_t = R(s_t, a_t, s_{t+1})$, and the transition function $P$ defines $P(s_{t+1}|s_t, a_t)$. Finally, we have the initial state distribution $\rho_0$ and the discount factor $\gamma$.

The goal of RL is to find an optimal policy $\pi^*$ that maximizes expected cumulative discounted return, with $\pi$ specifying action $a_t \sim \pi(\cdot \mid s_t)$ for each state $s_t$:

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

Value functions are ubiquitous in RL algorithms and provide the expected discounted return of a policy given an initial state $V^\pi(s)$ or a state-action pair $Q^\pi(s, a)$. Because the true value functions are usually unknown, they are often approximated, e.g., with neural networks parameterized by $\phi$. The state value $V_\phi^\pi(s)$ approximates the expected discounted return when following policy $\pi$ from state $s$:

$$V^\pi(s) = \mathbb{E}_\pi \left[ \sum_t \gamma^t R(s_t, a_t, s_{t+1}) \Big| \, s_0 = s \right]. \tag{2}$$

Analogously, $Q_\phi^\pi(s, a)$ estimates the expected discounted return for a state-action pair. LLM-TALE combines goal-conditioned state or state–action values with an uncertainty metric $c \in (0, 1]$ to trade off exploration and exploitation.

### B. Problem Formulation

We focus on pick-and-place manipulation tasks similar to those described in [34], [35]. The environment provides a sparse external reward $R^{\text{ex}} : S \times A \to \{0, 1\}$ that indicates task success. Our goal is to leverage the reasoning ability of LLMs to generate plans that guide exploration toward semantically meaningful regions of the state–action space, while a residual policy compensates for suboptimalities in the generated plans.

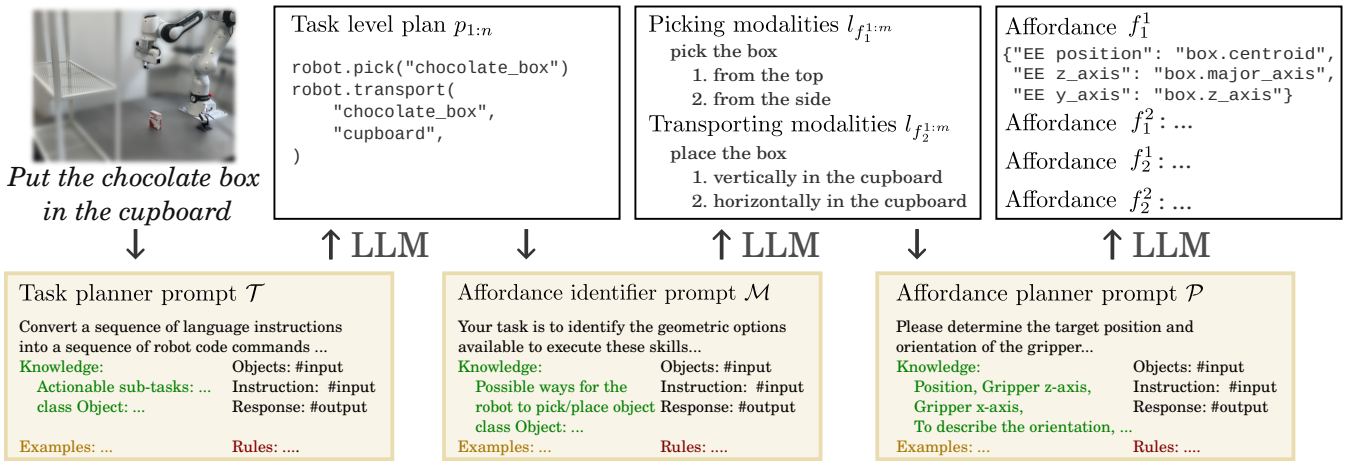



Fig. 2: Detailed visualization of the task planner scheme from Alg. 1, showing the structure of prompts $\mathcal{T}, \mathcal{M}$ and $\mathcal{P}$.

## IV. Method

### A. Overview of LLM-TALE

LLM-TALE consists of a planning phase and a training phase. Before training, the planning pipeline generates plans at the task and affordance levels. Task-level planning decomposes a language command into a sequence of $n$ primitives $p_{1:n}$ of two types: pick and transport. Affordance-level plans $\boldsymbol{f}_{1:n}$ describe the identified affordances and their affordance-level plans. We represent the complete plan as the ordered sequence $\boldsymbol{p} = \big(p_j(\boldsymbol{f}_j)\big)_{j=1}^{n}$, where $p_j \in \{\text{pick}, \text{transport}\}$. An overview of the planning process is presented in Fig. 1, while detailes are giving in Sec. IV-B and Fig. 2.

During training, the primitive $p_j$ translates the affordance-level plan into a goal $g_j \in SE(3)$ for the end-effector, conditioned on the objects' state $s^{\text{obj}}$. These goals are defined relative to an object pose, as in Fig. 1, where the two picking goals correspond to side and top grasps.

A hard-coded PD controller serves as a base policy, $a_p = \mu_{p_j}(s, g_j)$, and drives the end-effector from the current state toward $g_j$ along linear trajectories in position and orientation. The RL agent learns a residual action policy, $a_{\text{re}} \sim \pi(\cdot|s, g_j)$, as in [36]. This residual formulation steers exploration toward semantically meaningful regions online. The RL agent corrects inaccuracies and suboptimal behavior in LLM controllers, as depicted in Fig. 1. The executed action is the sum of $a_p$ and $a_{\text{re}}$:

$$a = a_p + a_{\text{re}}. \tag{3}$$

To guide exploration toward the goal, we define an intrinsic reward $r^{\text{in}}$. This dense reward is primitive-specific and requires no task-specific tuning:

$$r^{\text{in}} = R_j^{\text{in}}(s, g_j)\,. \tag{4}$$

Here, $R_j^{\text{in}}$ is a dense shaping term computed from pose errors relative to $g_j$ and the magnitudes of joint velocities.

Our LLM-guided initialization induces a semantically meaningful state distribution. As a result, the RL agent only refines the policy around the LLM-induced distribution, improving sample efficiency. The approach is compatible with both on-policy and off-policy algorithms and does not require (human) demonstration data.

### B. Task Planning

This section describes how the LLM converts high-level task instructions into task- and affordance-level plans. Our pipeline comprises three prompts: a task-level prompt $\mathcal{T}$, an affordance identifier prompt $\mathcal{M}$, and an affordance planner prompt $\mathcal{P}$. The prompt structure is inspired by [37], as our work targets similar setups.

As summarized in Alg. 1 and visualized in Fig. 2, the task-level prompt $\mathcal{T}$ translates a high-level task description $L$ into primitive action descriptions, such as "*Pick up the cube*" or "*Place the box in the cupboard*". Querying $\mathcal{T}$ with $L$ yields a sequence of Python primitives $p_{1:n}$ (e.g., `robot.pick(object)`) for reuse across episodes.

Affordance-level plans are often semantically multimodal, but not every mode is physically feasible. Consider placing a box in a cupboard (Fig. 1). The actions of picking and placing the box are ambiguous: it may be grasped from the top or the side and placed horizontally or vertically. However, placing an open box horizontally may spill its contents, while placing it vertically can make it unreachable for the robot. This example illustrates that an LLM-generated plan can be semantically valid yet physically infeasible without grounding in the real world. We use this scenario as a motivating example throughout the paper and refer to it as the *PutBox* experiment.

Because the LLM lacks the physical understanding required to directly select the best affordance, we introduce an affordance identifier prompt $\mathcal{M}$ that enumerates semantically distinct affordance plans when language admits multiple modes. For each primitive $p_j$, $\mathcal{M}$ produces language descriptions $l_{f_j^{1:m}}$ for $m$ affordance modalities, such as "*Pick the box from the side*" or "*Pick the box from the top*" (see Alg. 1).

The affordance planner prompt $\mathcal{P}$ then maps each description $l_{f_j^i}$ to a natural-language affordance $f_j^i$ that specifies the robot end-effector pose, rather than precise $SE(3)$ co-

**Algorithm 1:** Task Planning

**Input:** Task description $L$
**Prompts:** Task-planner prompt $\mathcal{T}$, modality-identifier prompt $\mathcal{M}$, affordance-planning prompt $\mathcal{P}$
**Output:** Task-level plans $p_{1:n}$, affordance-level plans $\boldsymbol{f}_{1:n}$

$p_{1:n} \leftarrow \texttt{query\_LLM}\big(\mathcal{T}(L)\big)$
**for** $j = 1 : n$ **do**
  $l_{f_j^{1:m}} \leftarrow \texttt{query\_LLM}\big(\mathcal{M}(p_j)\big)$
  **for** $i \leftarrow 1$ **to** $m$ **do**
    $f_j^i \leftarrow \texttt{query\_LLM}\big(\mathcal{P}(p_j, l_{f_j^i})\big)$
  $\boldsymbol{f}_j \leftarrow \{f_j^1, \ldots, f_j^m\}$
**return** $p_{1:n}$, $\boldsymbol{f}_{1:n}$

**Algorithm 2:** LLM-TALE

**Input:** Task description $L$
**Parameters:** Temperature $\beta$, update rate $\alpha$, minimum uncertainty $c_{\min}$, number of episodes $K$
**Output:** Policy $\pi$

$p_{1:n}$, $\boldsymbol{f}_{1:n} \leftarrow \texttt{plan\_task}(L)$ `// Alg. 1`
Initialize residual policy $\pi$, base policies $\mu_{1:n}$, value function $V_\phi^\pi$, uncertainties $\boldsymbol{c}_{1:n}$, time step $t \leftarrow 0$
**for** $k = 1 : K$ **do**
  $s_t, s_t^{\text{obj}} \leftarrow \texttt{reset()}$
  **foreach** $p_j \in p_{1:n}$ **do** `// Primitives`
    **foreach** $f_j^i \in \boldsymbol{f}_j$ **do** `// Affordances`
      $g_j^i \leftarrow \texttt{parse\_to\_goal}(s_t^{\text{obj}}, f_j^i)$
      $p_{\text{sel}}(i) \propto \exp\big(\beta\, V_\phi^\pi(s_t, g_j^i)\big)\, c_j^i$
    Sample $i \sim p_{\text{sel}}$ and set $g_j \leftarrow g_j^{(i)}$
    $c_j^{(i)} \leftarrow \max\big((1-\alpha)c_j^{(i)},\ c_{\min}\big)$
    **while** $p_j$ *not done* **do**
      $a_t \leftarrow \pi(s_t, g_j) + \mu_j(s_t, g_j)$
      $s_{t+1}, s_{t+1}^{\text{obj}}, r_t^{\text{ex}}, \text{done} \leftarrow \texttt{step}(a_t)$
      $r_t \leftarrow R_j^{\text{in}}(s_{t+1}, g_j) + r_t^{\text{ex}}$
      Store transition $(s_t, a_t, r_t, s_{t+1}, \text{done})$
      Update $\pi$ and $V_\phi^\pi$ with stored transitions
      $t \leftarrow t + 1$
**return** $\pi$

ordinates. For picking tasks, affordances are specified by three attributes: position, end-effector (EE) z-axis, and EE y-axis (the facing direction of the EE and the direction in which the gripper opens). This lets the LLM align robot motion with environmental features effectively. For transport tasks, we use object-centric formulations to accommodate discrepancies between expected and actual picking poses, improving robustness and environmental adaptability.

### C. Online Exploration

Our method explores affordance multimodality. During training, affordance plans $f_j^{1:m}$ are parsed into candidate goal poses $g_j^{1:m}$. For each primitive $p_j$, we score each goal $g_j^i$ with the value function and an uncertainty term, then sample $g_j$ from the resulting distribution. The value function estimates the expected return from the current state and thus approximates the utility of a goal under the present observation. For agents with a state-value function, we define the goal selection probabilities as

$$p_{\text{sel}}(i) \propto \exp\big(\beta\, V_\phi^\pi\big(s, g_j^i\big)\big)\ c_j^i, \tag{5}$$

and an analogous expression applies when using a Q-function $Q_\phi(s,a)$. Here, $\beta > 0$ controls the distribution's sharpness, and $c_j^i$ trades off exploration and exploitation. We then sample $i \sim p_{\text{sel}}$, set the goal $g_j \leftarrow g_j^{(i)}$, and update the corresponding uncertainty:

$$c_j^{(i)} \leftarrow \max\big((1-\alpha)c_j^{(i)},\ c_{\min}\big) \tag{6}$$

where $\alpha \in (0,1)$ and $c_{\min}$ is a lower bound on uncertainty. This multimodal affordance exploration strategy is visualized in Fig. 3. The complete procedure of LLM-TALE is described in Alg. 2.

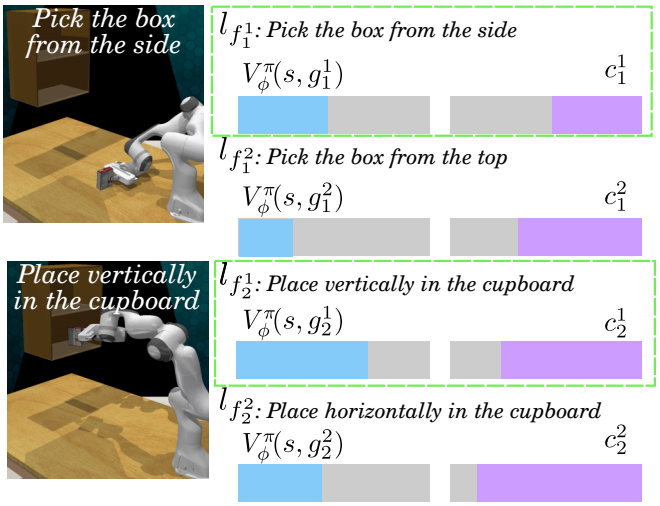



Fig. 3: LLM-TALE explores affordance modalities based on value $V_\phi^\pi(s, g_j^i)$ and uncertainty score $c_j^i$.

## V. Simulation Experiments

Our simulation evaluations span six tasks: three from RLBench [34] and three from ManiSkill [35][1]. The robot end-effector is controlled with relative position or velocity commands rather than joint-space control, which simplifies the action space for language models. The action is $a = (\delta_x, \delta_y, \delta_z, \delta_{rx}, \delta_{ry}, \delta_{rz}, \text{grip})$ and comprises translational and rotational deltas plus a gripper action. All experiments were repeated three times with different seeds.

We learn policies with a residual action space on top of base actions defined by pre-existing primitives (Alg. 2). These primitives are implemented with a PD controller that moves linearly toward the target. The proportional gain is set

[1]The corresponding code, hyperparameters and prompts are available at `https://github.com/llm-tale/llm_tale`.

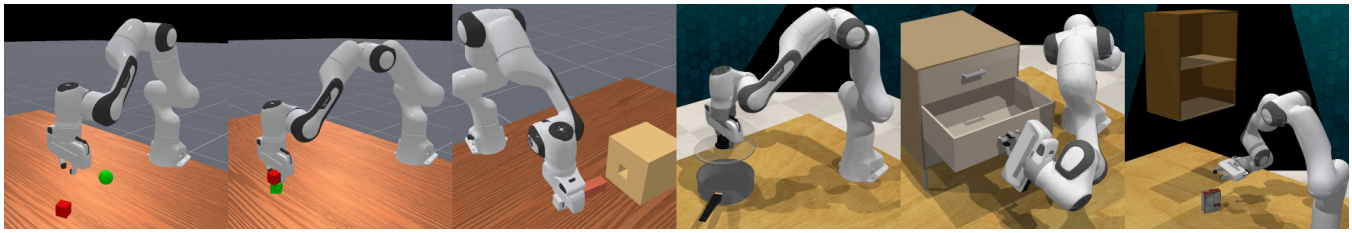


Fig. 4: Simulation tasks (left to right): *PickCube*, *StackCube*, *PegInsert*, *TakeLid*, *OpenDrawer*, and *PutBox*.

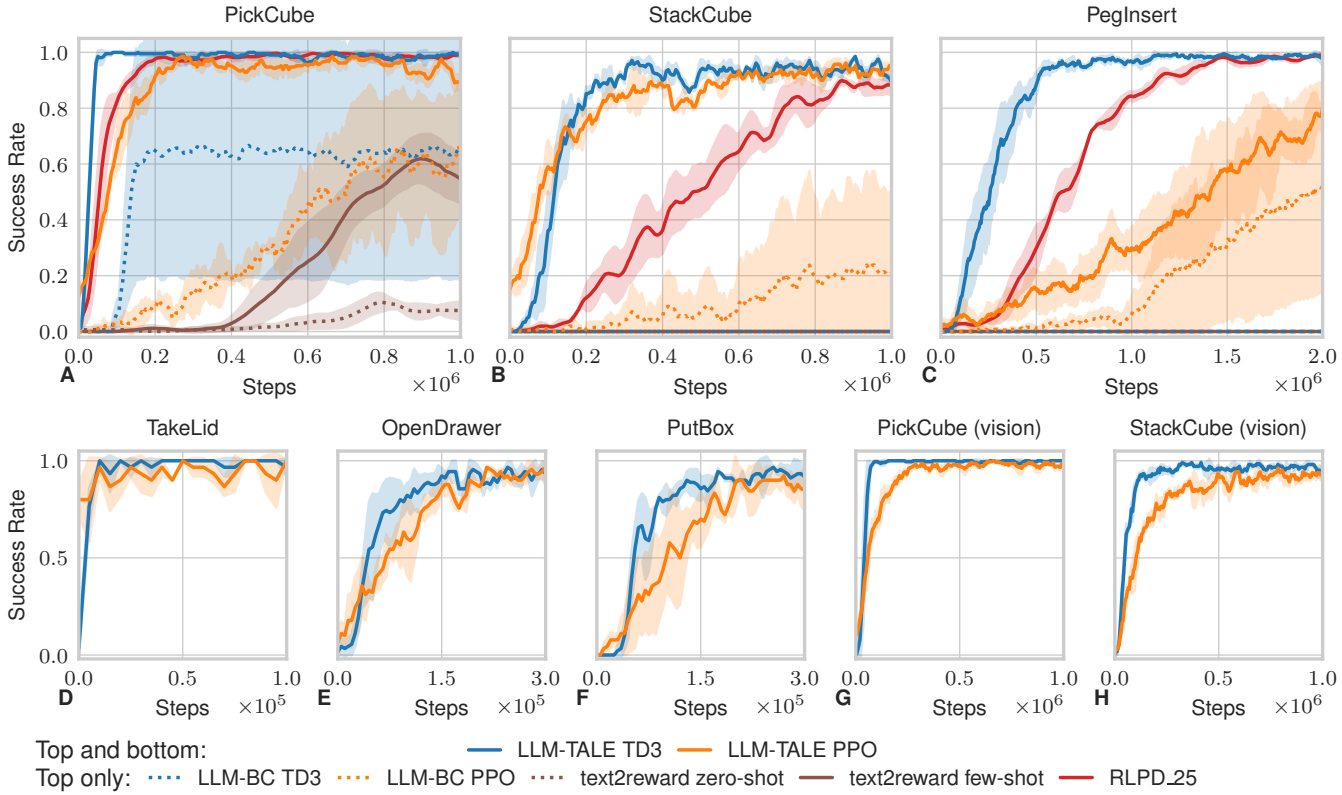



Fig. 5: Top figures (**A-C**) show evaluation results with comparisons against baselines and LLM-TALE ablations in ManiSkill [35] tasks, while bottom figures show evaluation results in RLBench [34] (**D-F**) and vision-based (**G-H**) tasks.

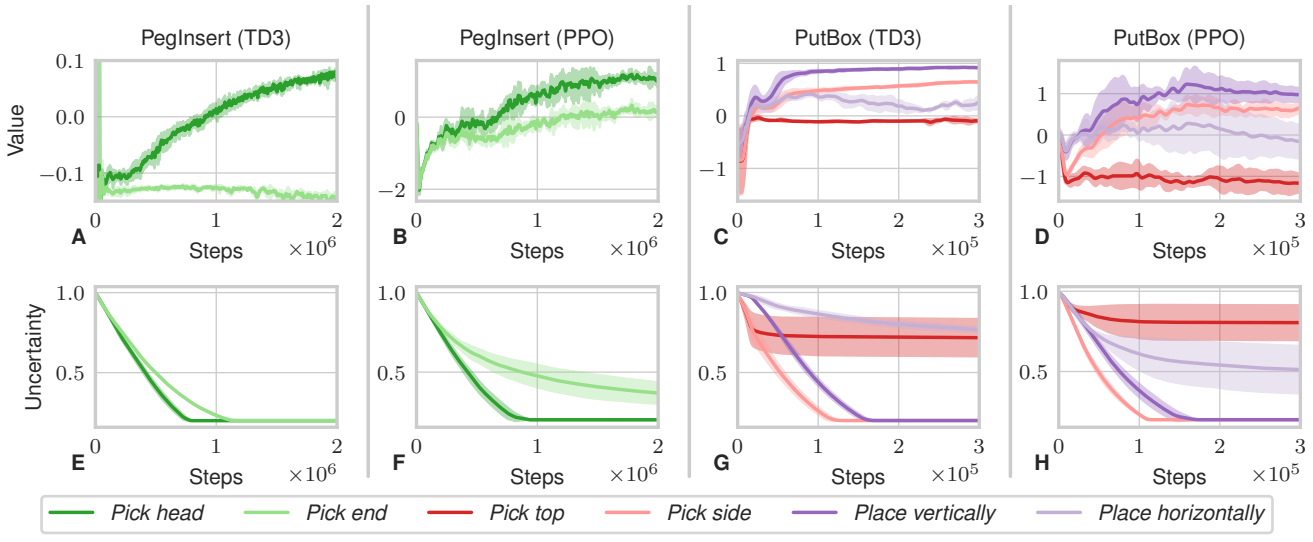



Fig. 6: Visualization of value- and uncertainty-based affordance exploration with LLM-TALE for *PegInsert* and *PutBox*.

to 1 by default, and motion is limited by maximum velocity and orientation-velocity bounds. For the *pick* primitive, the episode terminates when the object's displacement from its initial position exceeds a 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 use general dense intrinsic reward functions that apply across the primitive per benchmark suite. For example, for pick actions in the RLBench suite, the intrinsic reward is

$$r_j^{\text{in}} \;=\; -\tanh\!\big(5\,e_{\text{pos}}\big) \;-\; 0.25\,\tanh\!\left(\frac{\|\dot{q}\|}{\pi}\right). \qquad (7)$$

Here, $e_{\text{pos}}$ is the Euclidean distance between the end-effector and the goal position, and $\dot{q}$ denotes the robot joint velocities. In addition to the intrinsic reward $r^{\text{in}}$, the environment provides an external reward $r^{\text{ex}}$ (with $r^{\text{ex}} \gg r^{\text{in}}$) upon successful task completion or other termination conditions.

We use OpenAI GPT-4o [38] for planning, selected for its performance, cost, and latency. Each planner prompt includes a brief task description, specific instructions, and task-related knowledge. We also provide a few examples to specify the desired format and reasoning. Consistent with Alg. 2, we query the LLM before training and cache the outputs as Python functions and dictionaries for reuse during training.

### A. ManiSkill Tasks

We evaluate the LLM-TALE framework on three representative pick-and-place tasks from the ManiSkill suite [35]: *PickCube*, *StackCube*, and *PegInsert* (short for *PegInsertionSide*). These tasks are shown in Fig. 4. This experiment evaluates sample efficiency by comparing our method against three baselines and an ablation of LLM-TALE: Text2Reward [24] (zero-shot and few-shot), RLPD [10] with 25 high-quality demonstrations, and LLM-BC (ablation of LLM-TALE).

**Text2Reward** uses LLM reasoning to generate dense rewards, guiding robots through task steps and fostering meaningful behavior without expert reward examples or human fine-tuning [24]. There are two settings: zero-shot and few-shot. Zero-shot: the LLM sees only the task description and generic instructions (no task-specific examples). Few-shot: the prompt adds one example reward function to guide reward generation [2].

**RLPD** is an efficient RL method that leverages demonstrations, using $50\%$ offline data and 10 critics to prevent overfitting. We use the implementation from [39] and train with sparse rewards and 25 demonstrations.

**LLM-BC** is an ablation of LLM-TALE that replaces the PD controller with a behavior cloning (BC) base policy. This policy is trained on offline demonstrations generated by the PD controller following LLM-generated plans.

Fig. 5 A–C show the evaluation success rates for all methods. Our method directly guides the robot to LLM-generated goals (Alg. 2), yielding fast and stable convergence across all tasks with PPO [40] and TD3 [28]. The only exception is the PPO variant for the *PegInsert* task, which shows a positive trend yet underperforms the TD3 variant and RLPD_25. Although on-policy methods typically have lower sample efficiency than off-policy methods, this gap is reduced in our setting. This can be attributed to primitives that move toward LLM-generated goals, which provide effective actions that simplify policy optimization. Compared with RLPD_25 and Text2Reward, our approach is more sample-efficient in most settings. A further advantage over RLPD is that LLM-TALE does not require (human) demonstrations.

[2]Text2Reward is run with the authors' original ManiSkill2-based code, whereas all other results use ManiSkill3.

TABLE I: Tasks with multiple identified affordance modalities ($m > 1$). The remaining tasks have $m = 1$.

| Task | $m$ | Affordance Description |
|---|---|---|
| *PegInsert* | 2 | Pick the peg from the end or head. |
| *PutBox* | 4 | Pick the box from the side or top; place vertically or horizontally in the cupboard. |

In the LLM-BC ablation, the training strategy (rewards and exploration) is identical; only the base policy differs. The TD3 version achieves some success on *PickCube* but shows no improvement on the other tasks. The PPO on-policy variant shows a positive trend for all tasks but underperforms LLM-TALE. The BC base policy fails under out-of-distribution states, which conflicts with exploration during RL training and makes it less efficient than a stable PD-based policy.

We also evaluate vision-based variants on two tasks (Figs. 5G–H) to validate that our method can handle high-dimensional observations. Following DrQ-v2 [41], we use a similar encoder with random-shift augmentation and a CNN backbone for visual encoding. This setup attains high sample efficiency in on- and off-policy learning, comparable to state-based input.

### B. RLBench Tasks

RLBench [34] provides a wide range of everyday tasks, making it a suitable platform for evaluating our LLM task planning and affordance-level planning pipeline. We evaluate LLM-TALE on three additional tasks: *TakeLid*, *OpenDrawer*, and our motivating task *PutBox*. These tasks are also shown in Fig. 4. We use the control mode *EndEffectorPoseViaIK*, which issues relative position and orientation commands.

The *PutBox* task was implemented in RLBench by modifying the existing *PutGroceriesInCupboard* task. We remove other objects from the original scene and add a constraint that the robot must not tilt the box to avoid spilling its contents. This constraint is not specified to the LLM during planning; instead, we incorporate it as an external reward during RL training. We also include collision detection as an external penalty signal. The method explores both affordance and execution levels to learn a policy that places the box in the cupboard.

Results for these tasks are shown in Fig. 5 D–F and indicate high sample efficiency for TD3 and PPO variants.

### C. Affordance Exploration

Beyond task performance, we evaluate the affordance exploration mechanism of LLM-TALE. Table I reports the

outputs of the affordance-modality identifier. With the affordance identifier prompt $\mathcal{M}$, the LLM finds multimodal affordances for *PegInsert* and *PutBox*. For *PegInsert*, the LLM-generated plan includes picking the peg from the head or from the end. For *PutBox*, the identifier returns top and side picks and horizontal or vertical placements.

Affordance-exploration results are visualized in Fig. 6. In Fig. 6 A, the TD3 and PPO variants quickly learn that head grasps yield higher value than end grasps. Fig. 6 E shows that the agent still explores end grasps, but less frequently, as their uncertainty decays more slowly than for head grasps. The PPO variant shows similar trends but takes longer to separate the modalities. For *PutBox*, both TD3 and PPO eventually stop exploring top picks, since this choice hampers accurate placement without spillage (Fig. 6 G–H). Figs. 6 C–D show that both variants identify side picks and vertical placements as the highest-value modalities after approximately 50k steps.

## VI. Real-World Experiments

To assess real-world performance, we evaluate zero-shot transfer of a simulation-trained policy on the motivating task *PutBox* using a physical setup. In this experiment, we compare LLM-TALE with the LLM-generated linear base policy to evaluate the effectiveness of our residual learning approach.

### A. Experimental Setup

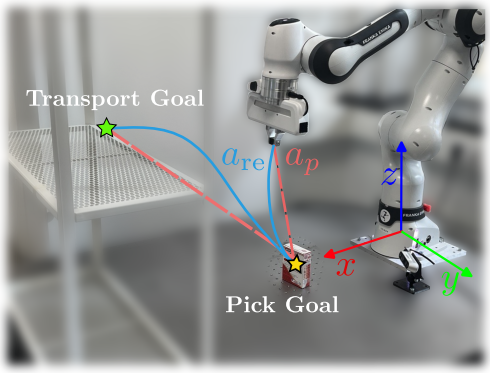
We used a Franka Emika Panda with a Franka gripper, controlled by a Cartesian impedance controller that commands the end-effector pose at 1 kHz. Our method sent end-effector pose commands to the impedance controller at a lower rate. We performed real-time object tracking using a RealSense D435 RGB-D camera, as shown in Fig. 7a.

### B. Results

We deployed a policy trained in simulation using LLM-TALE with TD3 on the robot and ran 15 episodes using our method and an LLM-only controller that directly executed LLM-generated plans. Our method achieved a success rate of $93.3\%$ with one failure, while the LLM-only base policy had a $0\%$ success rate due to collisions with the cupboard.

Figs. 7 A and 7 B show trajectories used to complete the task. The base primitive policy produces linear actions directed toward the goal pose identified by the LLM planner. Even without specific sim-to-real techniques, the primitive-based policy guides the robot toward higher-confidence regions when encountering unknown states.

The residual policy refines trajectories to ensure physical feasibility. The trajectory in Fig. 7 A shows that the RL agent learns to avoid collisions during picking. Moreover, the trajectory in Fig. 7 B shows that the residual policy increases vertical clearance during placement to avoid contact with the cupboard. The robot also executes larger, steadier motions, completing the task more efficiently than the LLM-only controller. Overall, the residual policy outperforms the LLM-only policy in both success rate and reliability.




(a) Real-world *PutBox* setup.

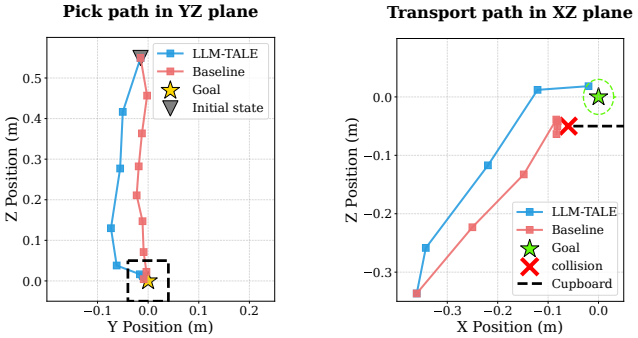



Fig. 7: Zero-shot sim-to-real experiment for the *PutBox* task.

## VII. Conclusion and Discussion

In this paper we introduce LLM-TALE, a framework for LLM-guided exploration at the task and affordance levels. We evaluate the approach in simulation against baselines based on LLM-guided reward shaping and sample-efficient reinforcement learning from demonstrations. LLM-TALE generates successful affordance-level plans, identifies multimodal solutions, and improves both success rate and sample efficiency over the baselines, even when only a single modality is discovered. The method is particularly suited for tasks with multiple affordances where the LLM lacks physical understanding. We demonstrate this in the motivating task *PutBox*, where the agent initially picks the box from the top but later learns to grasp it from the side, enabling successful placement without spillage. Real-world evaluations further demonstrate promising sim-to-real transfer.

Despite these advantages, the current planning framework does not handle objects with complex geometry and is limited to simpler items such as cubes, boxes, and drawer handles. The method also requires access to object state, which requires state estimators for real-world deployment.

To address these limitations, we plan to incorporate interactive learning in which a human provides one or a few demonstrations to help the robot plan for objects with more complex geometric relationships. To improve generalization, we also aim to use manipulation foundation models, such as AnyGrasp [42], to generate affordance goals without requiring precise object-state information.

## VIII. Acknowledgments

Research reported in this work was partially or completely facilitated by computational resources and support of the Delft AI Cluster (DAIC) [43] at TU Delft (RRID: SCR_025091), but remains the sole responsibility of the authors, not the DAIC team.

## References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, 2018.

[2] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Intl. J. Robotics Res. (IJRR)*, 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," in *NeurIPS Deep. Reinf. Learn. Workshop*, 2015.

[4] M. Bellemare, S. Srinivasan, G. Ostrovski *et al.*, "Unifying count-based exploration and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 29. Curran Associates, Inc., 2016.

[5] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," in *NeurIPS Deep. Reinf. Learn. Workshop*, 2017.

[6] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-based exploration with neural density models," in *Proc. Intl. Conf. on Mach. Learn. (ICML)*, vol. 70. PMLR, 2017, pp. 2721–2730.

[7] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Intl. Conf. on Mach. Learn. (ICML)*, vol. 70. PMLR, 2017, pp. 2778–2787.

[8] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Intl. Conf. on Robotics & Autom. (ICRA)*, 2018, pp. 6292–6299.

[9] A. Nair, M. Dalal, A. Gupta, and S. Levine, "AWAC: Accelerating online reinforcement learning with offline datasets," in *NeurIPS Deep. Reinf. Learn. Workshop*, 2020.

[10] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *Proc. Intl. Conf. on Mach. Learn. (ICML)*, vol. 202. PMLR, 2023, pp. 1577–1594.

[11] M. Vecerik, T. Hester, J. Scholz *et al.*, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint*, 2017.

[12] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation bootstrapped reinforcement learning," in *Proc. Robotics: Sci. Syst. (RSS)*, 2024.

[13] A. Bhaskar, Z. Mahammad, S. R. Jadhav, and P. Tokekar, "PlanRL: A motion planning and imitation learning framework to bootstrap reinforcement learning," *arXiv preprint*, 2024.

[14] W. Huang, X. Zheng, X. Ma *et al.*, "An empirical study of LLaMA3 quantization: from LLMs to MLLMs," *Vis. Intell.*, vol. 2, no. 1, p. 36, 2024.

[15] OpenAI, "GPT-4 technical report," *arXiv preprint*, 2023.

[16] S. Kambhampati, K. Valmeekam, L. Guan *et al.*, "Position: LLMs can't plan, but can help planning in LLM-modulo frameworks," in *Proc. Intl. Conf. on Mach. Learn. (ICML)*, vol. 235. PMLR, 2024, pp. 22 895–22 907.

[17] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proc. Intl. Conf. on Mach. Learn. (ICML)*, vol. 162. PMLR, 2022, pp. 9118–9147.

[18] B. Ichter, A. Brohan, Y. Chebotar *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," in *Proc. Conf. on Robot Learn. (CoRL)*, vol. 205. PMLR, 2023, pp. 287–318.

[19] A. Zeng, M. Attarian, B. Ichter *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," in *Proc. Intl. Conf. on Learn. Represent. (ICLR)*, 2023.

[20] W. Huang, C. Wang, R. Zhang *et al.*, "VoxPoser: Composable 3D value maps for robotic manipulation with language models," in *Proc. Conf. on Robot Learn. (CoRL)*, vol. 229. PMLR, 2023, pp. 540–562.

[21] J. Liang, W. Huang, F. Xia *et al.*, "Code as policies: Language model programs for embodied control," in *Proc. IEEE Intl. Conf. on Robotics & Autom. (ICRA)*, 2023, pp. 9493–9500.

[22] Y. Du, O. Watkins, Z. Wang *et al.*, "Guiding pretraining in reinforcement learning with large language models," in *Proc. Intl. Conf. on Mach. Learn. (ICML)*, vol. 202. PMLR, 2023, pp. 8657–8677.

[23] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," in *Proc. Intl. Conf. on Learn. Represent. (ICLR)*, 2023.

[24] T. Xie, S. Zhao, C. H. Wu *et al.*, "Text2Reward: Reward shaping with language models for reinforcement learning," in *Proc. Intl. Conf. on Learn. Represent. (ICLR)*, 2024.

[25] Y. J. Ma, W. Liang, G. Wang *et al.*, "Eureka: Human-level reward design via coding large language models," in *Proc. Intl. Conf. on Learn. Represent. (ICLR)*, 2024.

[26] R. Ma, J. Luijkx, Z. Ajanović, and J. Kober, "ExploRLLM: Guiding exploration in reinforcement learning with large language models," in *Proc. IEEE Intl. Conf. on Robotics & Autom. (ICRA)*, 2025, pp. 9011–9017.

[27] 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 Autom. Lett. (RA-L)*, vol. 9, no. 7, pp. 6075–6082, 2024.

[28] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Intl. Conf. on Mach. Learn. (ICML)*, vol. 80. PMLR, 2018, pp. 1587–1596.

[29] W. Huang, F. Xia, D. Shah *et al.*, "Grounded decoding: Guiding text generation with grounded models for embodied agents," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 36. Curran Associates, Inc., 2023, pp. 59 636–59 661.

[30] K. Lin, C. G. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2Motion: from natural language instructions to feasible plans," *Auton. Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[31] B. Liu, Y. Jiang, X. Zhang *et al.*, "LLM+P: Empowering large language models with optimal planning proficiency," *arXiv preprint*, 2023.

[32] Y. Chen, J. Arkin, C. Dawson *et al.*, "AutoTAMP: Autoregressive task and motion planning with llms as translators and checkers," in *Proc. IEEE Intl. Conf. on Robotics & Autom. (ICRA)*, 2024, pp. 6695–6702.

[33] M. Dalal, T. Chiruvolu, D. S. Chaplot, and R. Salakhutdinov, "Plan-Seq-Learn: Language model guided RL for solving long horizon robotics tasks," in *Proc. Intl. Conf. on Learn. Represent. (ICLR)*, 2024.

[34] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robotics Autom. Lett. (RA-L)*, vol. 5, no. 2, pp. 3019–3026, 2020.

[35] T. Mu, Z. Ling, F. Xiang *et al.*, "ManiSkill: Generalizable manipulation skill benchmark with large-scale demonstrations," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 1, 2021.

[36] T. Johannink, S. Bahl, A. Nair *et al.*, "Residual reinforcement learning for robot control," in *Proc. IEEE Intl. Conf. on Robotics & Autom. (ICRA)*, 2019, pp. 6023–6029.

[37] L. Zha, Y. Cui, L.-H. Lin *et al.*, "Distilling and retrieving generalizable knowledge for robot manipulation via language corrections," in *Proc. IEEE Intl. Conf. on Robotics & Autom. (ICRA)*, 2024, pp. 15 172–15 179.

[38] OpenAI, "Gpt-4o system card," OpenAI, Tech. Rep., 2024. [Online]. Available: https://openai.com/index/gpt-4o-system-card/

[39] S. Tao, A. Shukla, T. kai Chan, and H. Su, "Reverse forward curriculum learning for extreme sample and demo efficiency," in *Proc. Intl. Conf. on Learn. Represent. (ICLR)*, 2024.

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

[41] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Mastering visual continuous control: Improved data-augmented reinforcement learning," in *Proc. Intl. Conf. on Learn. Represent. (ICLR)*, 2022.

[42] H.-S. Fang, C. Wang, H. Fang *et al.*, "AnyGrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Trans. on Robotics (TRO)*, vol. 39, no. 5, pp. 3929–3945, 2023.

[43] Delft AI Cluster (DAIC), "The delft AI cluster (DAIC), rrid:scr_025091," 2024. [Online]. Available: https://doc.daic.tudelft.nl/