VLIN-RL: A Unified Vision-Language Interpreter and Reinforcement Learning Motion Planner Framework for Robot Dynamic Tasks

Zewu Jiang[1] Junnan Zhang[2] Ke Wang*[3] and Chenyi Si[4]

***Abstract*— Recently, with the development of Large Language Models (LLMs), Embodied AI represented by Vision-Language-Action Models (VLAs) has played a significant role in realizing the natural language interaction between humans and robots. Current VLA models can process and understand visual information and language instructions, while guiding robots to complete interactive tasks with the environment based on human language instructions. However, when tackling with the real-time and dynamic tasks, VLA has poor robustness and real-time planning and adjustment ability against changes in target objects, instructions, and environments. To handles these limitations, we propose VLIN-RL, a unified framework that consists of the Vision-Language Interpreter (VLIN) that owns excellent vision language information understanding and advanced task planning abilities and reinforcement learning (RL)-based motion planner with enhanced flexibility and broader applicability. If the environmental state changes during task execution, the RL planning module in VLIN-RL will directly make dynamic adjustments at the subtask level based on visual feedback to achieve the task goals, without the need for time-consuming information processing from VLIN. Experiments demonstrate that our model can complete multi-robot manipulation tasks more efficiently and stably. Finally, our work is verified by the pick-grasp tasks and real manipulators experiments.**

I. INTRODUCTION

In recent years, the emergence of LLMs has catalyzed a surge of embodied intelligence research. LLMs have demonstrated remarkable capabilities in reasoning, tool utilization, task planning, and instruction compliance, thereby unveiling novel possibilities for enhancing cognitive planning and decision-making capacities in multi-agent systems. VLAs are a type of multi-modal model in the field of embodied artificial intelligence, aimed at integrating and processing information from visual, language, and action modalities. For example, RT-2 [1] has recently been proposed to receive and process language instructions from humans and visual information from the environment, perform environmental scene understanding and task inference, and achieve task sequence planning to generate appropriate actions.

Benefiting from the increasing accuracy of visual encoders that help perceive target poses and environmental states [2-4] and the increasingly powerful information processing and logical reasoning capabilities of LLMs [1][5], VLAs can handle multi-process tasks in complex environments. Different VLAs have different focuses. In order to improve the performance of various robot tasks, some VLAs prioritize strengthening specific parts, such as high-quality pre

[1]Zewu Jiang, [2]Junnan Zhang, *[3]Ke Wang and [4]Chenyi Si are with the School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: jzw_2023234188@tju.edu.cn; zjn8018@tju.edu.cn; walker_wang@tju.edu.cn; sichenyi2021@163.com).

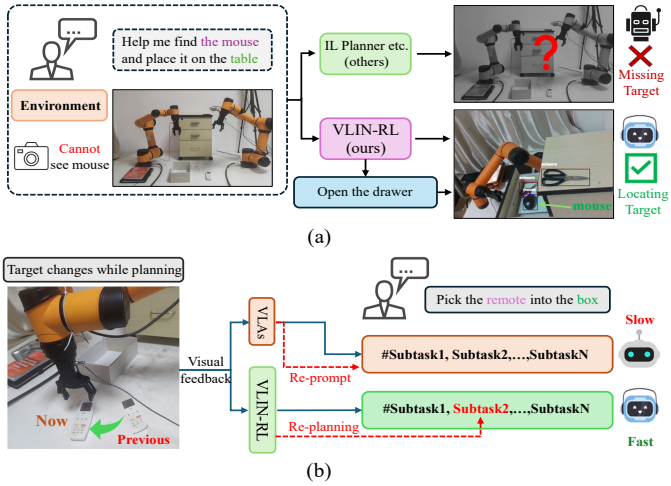

Fig. 1. (a) Tackle the multi-process task for finding invisible objects using different methods; (b) Tackle the real-time dynamic pick-place task using different methods.

trained visual representations (Contrastive Language-Image Pre-training, CLIP) [6], LLM-induced (DECKARD) [7], and Reasoning (Chain-of-Thought, CoT) [8-9]; Other VLAs focus on improving control strategies, specializing in receiving short-term task instructions and generating actions that can be executed through robot motion planning; (e.g., Early Non-Transformer Control Policies (CLIPort [2]), Transformer-based Control Policies (RT-1 [10]) and LLM-based Control Policies (RT-X [11]); In addition, some VLAs are detached from low-level control and focus on decomposing long-term tasks into subtasks that can be executed by low-level control strategies. Among these, the classic ones are Inner Monologue [12], ReAct [13], and LLM-Planner [14];

However, when facing real-time dynamic environments with complex constraints, VLAs cannot effectively provide real-time responsiveness, often resulting in the inability to complete tasks that require urgent response. Especially in some low-level motion planning, dynamic changes in the environment (such as changes in the pose of the target object) do not need to be responded to by LLM's time-consuming reprogramming. As is well known, reinforcement learning motion planning algorithms possess enhanced flexibility and broader applicability. When the coverage of train dataset is wide enough, RL methods have both faster response speed than VLAs and higher planning efficiency than traditional low-level motion planning strategies. However, when faced with complex multi-process tasks, RL methods lack the high-level task planning capabilities of VLAs based on

environmental understanding and logical reasoning.

In order to solve the dilemma of being unable to respond dynamically in a timely manner in complex multi-process tasks, we propose a novel framework VLIN-RL that combines Vision-Language Interpreter (VLIN) responsible for multi-process task planning with RL motion planning modules for timely response, as shown in Fig. 1. Meanwhile, to test the effectiveness of our proposed framework, we designed several complex tasks with multi-step and dynamic changes. Besides, in order to ensure the absolute correctness of the planning made by VLIN, we introduced PDDL [15] and the state-of-the-art symbolic planner called Fast Downward [16] to verify the correctness of the planning order from LLMs and rectify the mistakes in the form of feedback.

Our contribution consists of the following four parts:

- Combine the LLM with PDDL and symbolic planner to verify the correctness of the planning order from LLMs, which named VLIN.
- VLIN-RL, a unified framework which bridges VLIN and RL motion planner to handle the complex multi-step tasks with dynamic changes.
- The WorkSpace-Reachable (WsReach) dataset used to train RL motion planning algorithms, a new dataset that covers all workspaces of the robotic arms, ensuring that the RL motion planner can handle a wide range of planning tasks.
- Several experiments to validate the effectiveness of the proposed VLIN-RL, a task set which contains complex multi-process tasks with dynamic changes.

II. RELATED WORK

This section describes both the related work of VLA, RL motion planning, symbol planning, and the comparison of performance with different methods in completing robotic interaction tasks, as shown in Fig. 2

A. Vision-Language-Action Models

In the field of embodied intelligence, VLA models typically process visual and linguistic instruction information and decompose tasks into subtask sequences as the high-level task planners. Among these, several task planners are built on the basis of LLM, incorporating training with multi-modal data to form the end-to-end VLA models [17-19]. Although the end-to-end VLAs models can simplify the process of handling multi-modal information, multi-modal training consumes massive computational resources. Consequently, the language-based VLA models construct the high-level task planner by converting multi-modal information into the form of language that LLM can process [20]. However, these VLA task planners have two drawbacks. First, during task execution, if the state of the target object changes, the VLA task planner cannot perform timely replanning due to the delay of LLM. Second, the planned order of task execution cannot guarantee complete correctness. VLIN-RL can solve the above problems by combining VLA high-level task planners and RL motion planning methods which can response the changes from environment in time. Besides, we introduced PDDL and the state-of-the-art symbolic planner called Fast Downward [16] to verify the correctness of the planning order from LLMs and rectify the mistakes in the form of feedback.

B. RL motion planning

Although control policies demonstrate proficiency in processing and implementing basic linguistic commands, they frequently encounter challenges when addressing complex, multi-step objectives requiring sequential execution of interrelated components. To overcome these limitations, the motion planning algorithm for robots which is used as the method to complete subtasks need the combination with high-level task planner in VLAs. Traditional motion planning algorithms are composed by four groups including graph search algorithms (e.g., A*) [21-23], sampling-based algorithms [24-26] like rapidly-exploring random tree (RRT), interpolating curve algorithms [27-29] (e.g., line and circle), and reaction-based algorithms [30] (e.g., potential field method (PFM)). However, these algorithms are all difficult to meet the success rate and efficiency at the same time.

In recent times, fueled by the rapid advancements in artificial intelligence, deep reinforcement learning (DRL) has showcased its adeptness in learning complex data patterns across a range of decision-making domains, including computer vision [31], gaming [32], manipulator operations [33], and motion control [34]. The RL motion planning algorithm takes the real-time state of the robot as input to the decision function, which outputs the corresponding action selection. Therefore, when the real-time state in the environment changes, RL planner can quickly complete replanning to adapt to the changes in the environment.

C. Symbolic Planning with PDDL

Symbolic planning (automated planning) has been widely employed in robotic task execution frameworks [35]. Symbolic planners enhance robotic transparency through human-readable domain models (with intuitive variables) and logically verified plans. These merits position their integration with language-guided methods as pivotal for developing interpretable robots. The descriptions are written in formal languages, such as PDDL [15] and PDDLStream [36]. The Vision-Language Interpreter of VLIN-RL is functionally integrated with such PDDL-driven architectures to mediate the conversion of natural language commands into structured problem definitions (PDs), bridging high-level linguistic inputs with symbolic planning systems.

III. PROBLEM STATEMENT

A. Vision-Language Interpreter

We consider an office scenario with dual robotic arms, which are responsible for finding invisible objects or rearranging targets. This requires VLIN to perform task decomposition and logical reasoning, forming multiple subtasks that RL can execute. The input of VLIN comes from natural language instructions from humans and environmental information from camera images. The output is a problem

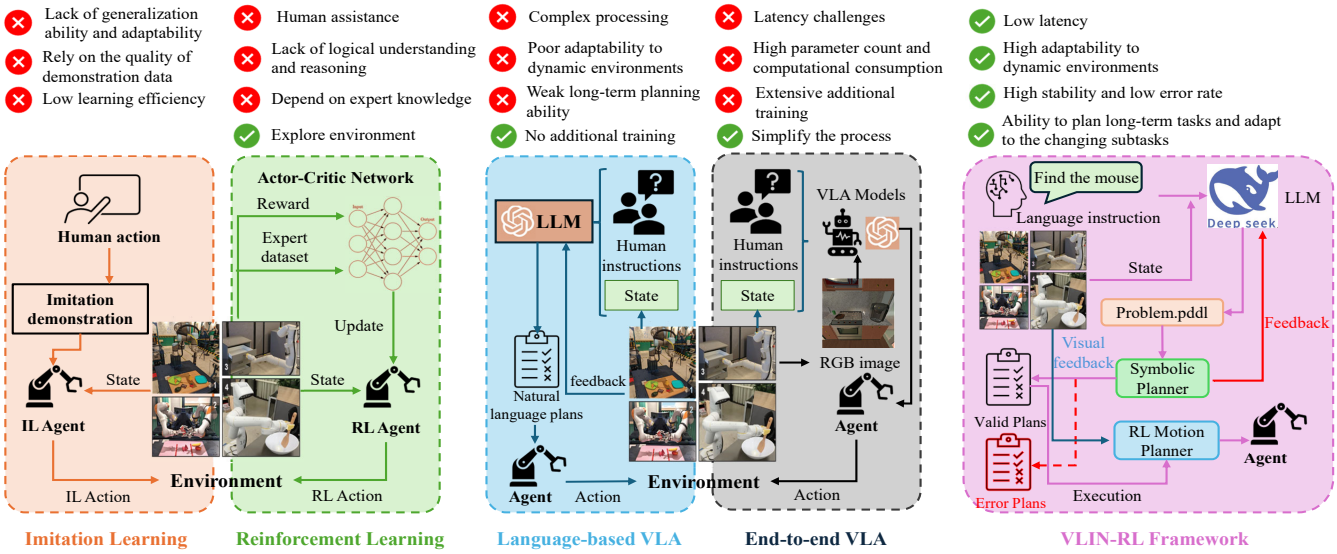

Fig. 2. Comparative Overview of different methods for robotic interaction tasks. (a) Core principles and operational limitations of IL/RL-based methods for robot planning policy learning. (b) Core principles and operational limitations of LLM-based methods for robot interaction task. (c) Our proposed VLIN-RL model, can handle the complex multi-step tasks with dynamic changes.

description consisting of $(\boldsymbol{O}, \boldsymbol{I}, \boldsymbol{G})$: the objects $\boldsymbol{O}$, the initial state $\boldsymbol{I}$, and the goal specification $\boldsymbol{G}$. The problem description can be parsed by the symbol planner into a sequence of subtasks while ensuring absolute correctness of the task execution order.

B. Reinforcement Learning for Motion Planning

In RL, Markov Decision Process (MDP) serves as a fundamental framework for modeling sequential decision-making problems. An MDP encapsulates key components within a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces respectively. The state transition probability function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ quantifies the likelihood of moving from state $\boldsymbol{s}_t$ to the subsequent state $\boldsymbol{s}_{t+1}$ upon taking action $\boldsymbol{a}_t$. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns numerical rewards, while the discount factor $\gamma \in [0, 1]$ determines the importance of future rewards.

The policy denoted by $\pi_\theta(\boldsymbol{a} \mid \boldsymbol{s})$, indicates the probability of executing action $\boldsymbol{a}$ at state $\boldsymbol{s}$. $\rho : \mathcal{S} \to [0, 1]$ gives the occurrence probability of the initial state $\boldsymbol{s}_0$. The trajectory $\tau = \{\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots\}$ represents an episode. In an MDP, the probability of a trajectory can be defined as

$$P_{\pi_\theta}(\tau) = \rho(\boldsymbol{s}_0) \prod_{t=0}^{T} \pi_\theta(\boldsymbol{a}_t \mid \boldsymbol{s}_t) P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t). \quad (1)$$

In an MDP, the objective is to discover a policy π_θ that maximizes the expected cumulative return

$$J(\theta) = \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(\tau)], \quad (2)$$

where $R(\tau) = \sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Therefore, the gradient of $J(\theta)$ can be written as

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim P_{\pi_\theta}}[R(\tau)] \\ &= \mathbb{E}_{\tau \sim P_{\pi_\theta}}[\nabla_\theta \log P_{\pi_\theta}(\tau) R(\tau)] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_{\pi_\theta}(\tau_i) R(\tau_i), \end{aligned} \quad (3)$$

where the i indicates the ith trajectory. The state value function $V^\pi(\boldsymbol{s})$ and the state-action value function $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ are utilized to assess the performance of the policy π. They are defined as

$$V^\pi(\boldsymbol{s}) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \mid \boldsymbol{s}_0 = \boldsymbol{s} \right], \quad (4)$$

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \,\middle|\, \begin{matrix} \boldsymbol{s}_0 = \boldsymbol{s} \\ \boldsymbol{a}_0 = \boldsymbol{a} \end{matrix} \right]. \quad (5)$$

IV. METHOD

The VLIN-RL framework is mainly divided into four parts: scene object detection, visual-language interpreter, symbolic planner, and reinforcement learning motion planner. The operational mechanism underlying the novel architecture is shown in Fig. 3.

A. Scene target perception

The scene perception part is responsible for detecting target objects in the environment and locating their 3D coordinate positions through camera calibration. Meanwhile, the camera will transmit the target coordinates and grasping pose in real time to the RL motion planner, ensuring that once the target object state of the subtask changes, the RL planner can perform real-time re-planning. We need to detect different objects in the scene to prevent any objects

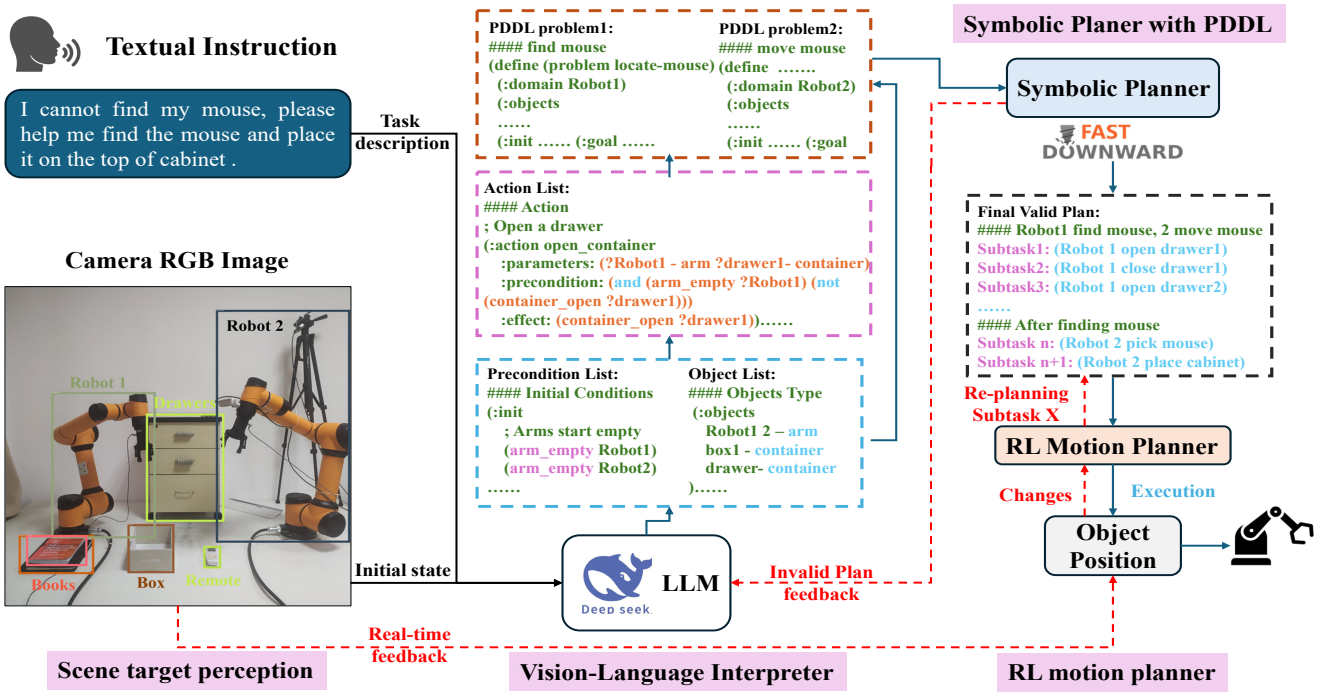

Fig. 3. Fundamentals and schematic representation of our proposed VLIN-RL framework. The VLIN-RL framework consists of four parts: a) scene target perception, Visual-Language Interpreter, Symbolic Planner with PDDL, and Reinforcement Learning motion planner.

from not being recognized. For this reason, we adopted the GroundingDINO [37], the SOTA open set object detection algorithm. Besides, we used the Grasp-Anything [38] method for grasping pose estimation to ensure that the robotic arm can successfully grasp the target pose when it changes.

B. Vision-Language Interpreter

VLIN consists of two parts: DeepSeek R1 as the LLM and Symbol Planner with PDDL. DeepSeek generates problem definitions PD including goal description G and domain knowledge with the executable actions based on the initial state I of the environment from camera visual information and natural language instructions provided by humans. This requires LLM for task understanding and logical reasoning. The goal description G must represent the instructions in human natural language.

In the office scene, we propose a task of finding something and placing it in a designated location. But the target object is invisible, which requires LLM to infer based on environmental information (such as the target object may be in a closed container or camera blind spot), and then plan subtasks to find the target object (opening drawers or exploring blind spot environments).

C. Symbolic Planer with PDDL

PDDL is a formal language designed for automated planning, enabling the specification of planning domains (environments) and planning problems (tasks) in a structured, machine-readable format. The symbolic planner inputs the problem definition and domain knowledge output by LLM for parsing, and outputs the subtask execution sequence for the RL motion planner to execute one by one.

The reason why we use PDDL instead of letting LLM directly output subtask sequences is to prevent errors in the planning order of LLM. PDDL uses symbols as a medium for objects and variables in planning, which ensures that errors do not occur during the planning process (such as when the robotic arm is grasping an object and LLM plans the robotic arm to open the drawer, ignoring the fact that the gripper is still closed).Meanwhile, PDDL also has the characteristic of human readability, which increases the interpretability of LLM.

D. RL motion planner

Due to the RL motion planner taking over the low-level control of all robotic arms, the RL planner must be able to plan for the robotic arm to reach all places within its workspace. For this reason, we train the RL planner with a new dataset that covers all workspaces of the robotic arms, ensuring that the RL planning method can handle a wide range of planning tasks. If the environmental state or target object changes during subtask execution, it is unnecessary and time-consuming for VLIN to re-plan to adapt to the environmental changes. We only need to feed back the camera visual information that captures the changes to the RL planner which can change the output planning action in real time according to the changing input state based the trained actor policy.

V. DATASET AND TRAINING

Due to the robotic high degrees-of-freedom (DoF), large-scale continuous action space and tightly coupled workspaces, particularly in scenarios where rewards are scarce, the exploration efficiency for multi-goal tasks is typically low. To tackle these issues, we adopt the Muti-Agent Soft-Actor-Critic (MA-SAC) algorithm [41], which currently has the better trade-off between exploration and exploitation in continuous action spaces, as the reinforcement learning algorithm for RL motion planners. The training framework is shown in Fig. 4.

In motion planning tasks, the next action of the arm is often closely linked to both current and history state. However, in the multi-arm systems, the training will be costly and inefficient for high dimensional state space including both current and history state. To handle the problem, we propose to use a LSTM [39] as the state encoder to extract a fixed-sized state sequence of each arm from the high dimensionality of current and history state. The extracted fixed-sized state sequence is a 256-dimensional embedding denoted as s^t.

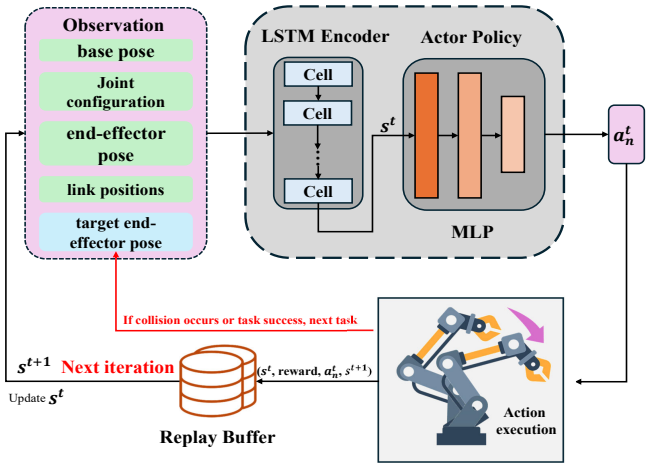

Fig. 4. The training framework of dual-arms RL motion planner.

A. Dataset for training

To ensure that the RL planner can control the robotic arm to reach all ranges of its workspace, the task dataset used for training must cover the entire workspace. We conduct simulation experiments on multi-arm motion planning tasks in the shared workspace with the PYBULLET [40] engine for training dataset. The 6-DoF robotic arms named AUBO i5 are used in the simulation. The motion planning task dataset are produced following an identical procedure as described below.

For the planning task including n arms, each arm sets to random base poses and randomly select target joint configurations within the working range of each robotic arm that will not collide. Then the target end-effector position of the task is calculated through target joint configurations by forward kinematics (FK). The tasks generated by this method can ensure the existence of collision-free planning results and coverage of the entire workspace so that RL motion planner can explore and learn collision-free planning paths. We collected motion planning tasks data for two AUBO i5 arms. The training dataset consists of more than 1,000,000 tasks, and the testing dataset consists of around 30,000 tasks.

B. Training Details

Observation: Each arm's state encompasses its base pose, joint configuration, and target end-effector pose. To facilitate the learning process, we enhance each arm's state by incorporating its end-effector pose, link positions, and including a single frame of past histories for all state components except the base pose.

Training Procedure: In the article, the motion planner is trained based on the SAC algorithm. The training process consists of episodes. In each episode, a new random task is chosen as the goal, and all robotic arms are initialized based on the initial position provided by the task information. The actor policy input the observation from the environment and output the corresponding action values which make the robot arms move in the simulation physics engine. Then as the state vectors, action values, and reward values of the robotic arms are placed into the experience replay buffer, the Critic networks of policy will update to fit the expected reward value and the policy loss function is used to update Actor networks.

The episode ends if any arm collides, if all arms successfully reach their targets, or if the episode duration reaches 300 time steps. The success of task is defined as the simultaneous achievement of target positions by all arms; failure occurs if this condition is not met.

Reward Function: To accelerate convergence, we design additional reward function for the policy. The policy will be rewarded or penalised in the different cases, as follows:

$$R^L = \begin{cases} r^L_{\text{reach}}, & \text{if } d_t \le \delta^L, \\ r^L_{\text{collision}}, & \text{if collision}, \\ r^L_{\text{overtime}}, & \text{if } t_L \ge T, \\ r^L_{\text{approach}}, & \text{if approaching the goal}, \\ r^L_{\text{rotate}}, & \text{if rotate over}. \end{cases} \tag{6}$$

r^L_{reach} is positive when the arm reaches the target position, i.e., when its distance to the target position, d_t, is within δ^L representing the minimum allowable error; $r^L_{\text{collision}}$ and r^L_{overtime} are the penalty when the policy can not complete the planning task within the set time; r^L_{approach} is positive reward when the robot is approaching the target, or negative punishment when getting away; r^L_{rotate} represents the orientation errors.

Training Evaluation: A task is considered to be successful if all arms reach the target end-effector poses within the distance of 0.02m and the orientation of 0.1 radians. The strategy network with a success rate of 95% after training will be used as the RL motion planner.

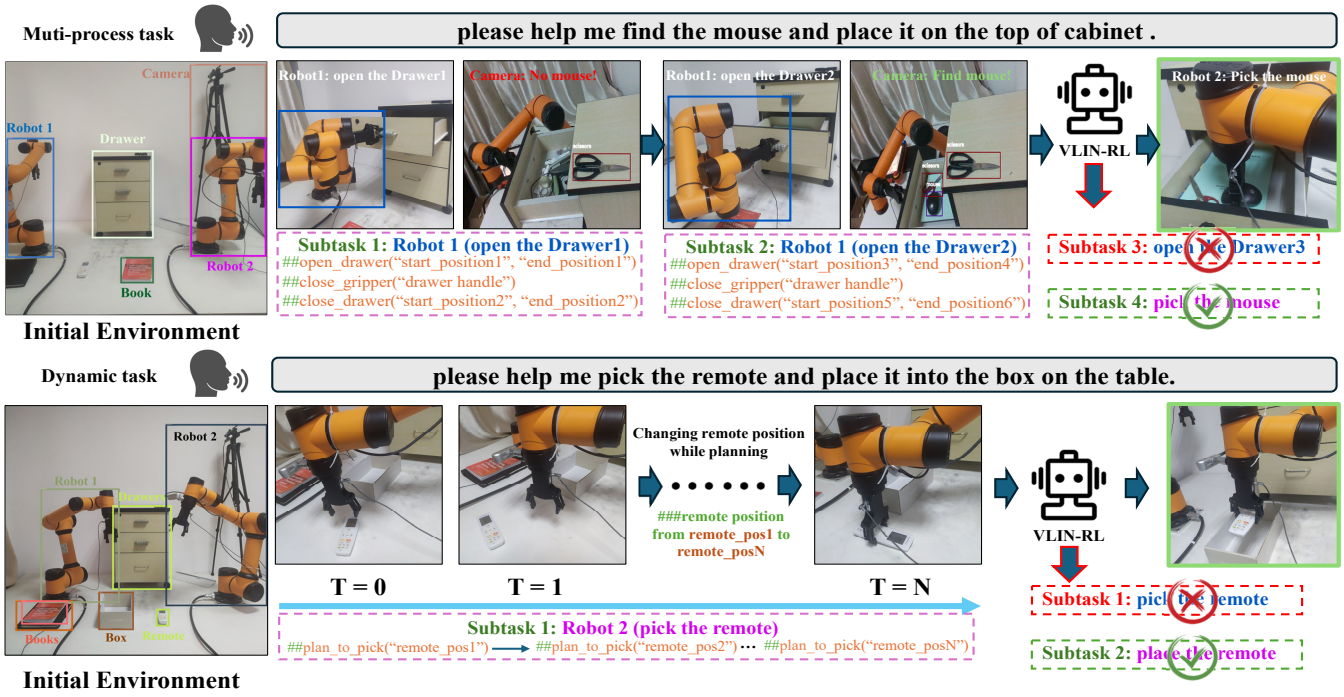

Fig. 5. The keyframes illustrate the execution of two tasks in different situation within the VLIN-RL framework. For the first multi-process task , after each subtask is completed, the camera located above the space provides timely feedback on whether it has detected the visual information of the target object (mouse) to ensure accurate adjustment of the next subtask. For the second dynamic task, the camera must provide real-time feedback during the execution of subtasks.

VI. EXPERIMENTS

The experimental part will be divided into three parts: a) Sim2Real experiment of RL motion planner to demonstrate the ability of VLIN-RL in real robotic arm motion planning. b) Vision-language-guided multi-process task testing experiment to show that VLIN-RL can analyze and complete long-term complex tasks with good performance. c) Evaluate the performance of VLIN-RL in real-time dynamic task experiments, as shown in Fig. 5.

In the real world, we tested the performance of the proposed framework using two AUBO i5 robotic arms equipped with three cameras simultaneously, including two calibrated RealSense D435i cameras fixed on the end effector of the robotic arm and above the space, and one Kinect2 camera. The end effectors of both robotic arms are equipped with INSPIRE-EG2-4B grippers. We used the recently released Deep Seek R1 model for the LLM part of VLIN-RL, which has excellent task understanding and logical reasoning capabilities.

A. Sim2Real for RL Motion Planner

To demonstrate the practicality and feasibility of Sim to Real experiment based on the trained motion planner model, we will complete the Pick-Place task in both simulation and real environment. For the simulation demonstration, we utilize PyBullet physics engine along with 6-DoF AUBO i5 robots equipped with two grippers. To achieve sim to real, we use the trained RL strategy network as the planner for the robotic arm. The target position and planning accuracy

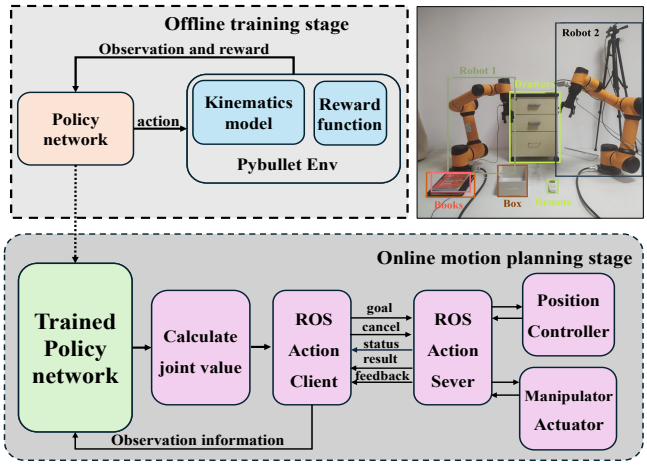

Fig. 6. The overall framework of Sim2Real for RL motion planner.

of the robotic arms are manually provided. The joint angle values of the real robotic arms are measured by sensors which have very small measurement errors. In the process of motion planning, the end effector positions and link positions of the robotic arms are the input of the planner which are calculated through forward kinematics by joint values. The planner's output corresponds to the change in joint angles. The communication between all the information obtained by these sensors and joint control commands are established within the robot operating system (ROS framework), as shown in Fig. 6.

TABLE I

COMPARISON RESULTS OF DIFFERENT METHODS IN MULTI-STEP TASK USING TIME CONSUMPTION (s) AND SUCCESS RATE (%) WITH THE CORRESPONDING STANDARD DEVIATION OVER 10 RUNS

ENV (Sim/Real)	VLIN-RRT	RL-Only	VLA-Only	Ours
Sim	3.2±0.1/87.9%	1.5±0.1/42.5%	13.8±0.3/85.3%	**2.4±0.1/90.8%**
Real	5.5±0.3/78.1%	2.6±0.1/30.3%	16.7±0.3/74.8%	**3.2±0.1/82.7%**
Collision	10.8%	7.6%	18.3%	2.3%
Pos-error	0.04±0.01	0.02±0.01	0.06±0.01	0.01±0.01

TABLE II

COMPARISON RESULTS OF DIFFERENT METHODS IN DYNAMIC TASK USING TIME CONSUMPTION (s) AND SUCCESS RATE (%) WITH THE CORRESPONDING STANDARD DEVIATION OVER 10 RUNS

ENV (Sim/Real)	VLIN-RRT	RL-Only	VLA-Only	Ours
Sim	3.6±0.1/87.6%	2.3±0.1/72.5%	12.7±1.5/82.5%	**1.6±0.1/95.3%**
Real	5.8±1.3/76.1%	4.6±0.3/60.3%	14.9±1.8/68.8%	**2.5±0.1/87.1%**
Collision	3.6%	2.1%	8.3%	1.1%
Pos-error	0.04±0.01	0.02±0.01	0.06±0.01	0.01±0.01

Since the measurement error of the joint sensors in an actual manipulator is minimal, and there is a negligible difference in the kinematic parameters between a real robot and its simulated counterpart, the error in the manipulator positions information obtained through forward kinematics calculation based on angle information will also be very small. The aforementioned factors facilitate the direct transfer of the policy network, which is trained in a simulated environment, to the real environment.

B. Vision-language-guided multi-process tasks in real-world

To verify the excellent logical reasoning and task decomposition of the proposed VLIN-RL framework, we designed a multi-process task for finding invisible objects. To complete this task, VLIN-RL needs to reason and decompose the total task into subtask actions of finding the target object, and adaptively change the next subtask based on the results of each subtask completion (whether the target object is found).

In this experiment, we used the RGB image of the Kinect2 camera as the initial environment. When planning the execution of actions, the RealSense D435i camera located above the space can observe the interior space of the drawer and provide real-time feedback of RGB image visual information for real-time detection and localization of target objects.

In order to ensure that the PDDL problem definition and domain knowledge output by VLIN-RL are completely correct, we have set up a symbolic planner (Fast Downward) to parse and plan subtask plans. If errors are found during this process (such as PDDL symbol errors or planned actions that cannot be implemented in the real world), the symbol planner will feedback the errors to LLM, which will re-plan until the plan is completely correct.

When executing subtasks, the trained RL motion planner only needs to accept the current pose and target position of the robotic arm to plan the next action with extremely small time delay, which is more efficient than traditional planning algorithms. The experimental results indicate that the VLIN-RL we proposed has successfully completed the multi-process task for finding invisible objects. The experimental data results are shown in TABLE I.

C. Real-time dynamic tasks in real-world

In order to test whether the VLIN-RL framework has the ability to adapt in real-time during task execution, we designed a task for a robotic arm to grasp and place dynamic objects. The target objects to be grasped and the destination to be placed in the task can be dynamically changed.

In this real-world experimental test, we selected the remote and box as target object and destination respectively. During the execution of the subtask, we achieved dynamic changes in the positions of the target object and destination by manually moving the remote and box. The experimental results demonstrate that VLIN-RL has a small time delay when processing dynamic tasks (as shown in the attached video). The experimental data results are shown in TABLE II

From the experimental results, we can see that when the position or destination of the target changes dynamically, VLIN-RL can use the RL motion planner to make dynamic adjustments at the subtask level, which saves a lot of time compared to LLM replanning and greatly improves real-time response capability.

VII. CONCLUSIONS

In this work, we address the critical challenges of real-time adaptability and robustness in Vision-Language-Action (VLA) models for dynamic robotic tasks. By proposing the VLIN-RL framework, which integrates a Vision-Language Interpreter (VLIN) with reinforcement learning (RL)-based motion planning, we bridge the gap between high-level language-conditioned task understanding and real-time reactive control.

While VLIN-RL demonstrates promising results, scaling the framework to more complex tasks (e.g., long-horizon manipulation with tool use) and integrating multi-agent coordination mechanisms remain open challenges.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB1714700 and in part by the National Natural Science Foundation of China under Grant 62333016.

REFERENCES

[1] A. Brohan, N. Brown, H. J. Carbajal, and *et al.*, "RT-2: vision-language-action models transfer web knowledge to robotic control," *Proceedings of The 7th Conference on Robot Learning (CoRL)*, vol. 229, pp. 2165–2183, 2023.

[2] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," *Proceedings of The 5th Conference on Robot Learning (CoRL)*, vol. 164, pp. 894–906, 2021.

[3] I. Radosavovic, T. Xiao, S. James, and *et al*., "Real-world robot learning with masked visual pre-training," *Proceedings of The 6th Conference on Robot Learning (CoRL)*, vol. 205, pp. 416–426, 2022.

[4] A. Majumdar, K. Yadav, S. Arnaud, and *et al*., "Where are we in the search for an artificial visual cortex for embodied intelligence," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 655-677, 2023.

[5] C. Raffel, N. Shazeer, A. Roberts, and *et al*., "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[6] A. Radford, J. W. Kim, C. Hallacy, and *et al*., "Learning transferable visual models from natural language supervision," *International Conference on Machine Learning (ICML)*, vol. 139, pp. 8748–8763, 2021.

[7] K. Nottingham, P. Ammanabrolu, A. Suhr, and *et al*., "Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling," *International Conference on Machine Learning (ICML)*, vol. 202, pp. 26311–26325, 2023.

[8] T. Kojima, S. S. Gu, M. Reid, and *et al*., "Large language models are zero-shot reasoners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 22199–22213, 2022.

[9] J. Wei, X. Wang, D. Schuurmans, and *et al*., "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 229, pp. 24824–24837, 2022.

[10] A. Brohan, N. Brown, H. J. Carbajal, and *et al*., "RT-1: robotics transformer for real-world control at scale," *Robotics: Science and Systems*, 2023.

[11] O. X. Collaboration, A. Padalkar, A. Pooley, and *et al*., "Open x-embodiment: Robotic learning datasets and RT-X models," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024.

[12] W. Huang, F. Xia, T. Xiao, and *et al*., "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[13] S. Yao, J. Zhao, D. Yu, and *et al*., "React: Synergizing reasoning and acting in language models," *International Conference on Learning Representations (ICLR)*, 2023.

[14] C. H. Song, B. M. Sadler, J. Wu, and *et al*., "LLM-planner: Few-shot grounded planning for embodied agents with large language models," *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, pp. 2998–3009, 2023.

[15] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *ournal of artificial intelligence research*, vol. 20, pp. 61124, 2003.

[16] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[17] B. Ichter, A. Brohan, Y. Chebotar, and *et al*., "Do as I can, not as I say: Grounding language in robotic affordances," *Proceedings of The 6th Conference on Robot Learning (CoRL)*, vol. 205, pp. 287–318, 2022.

[18] S. Li, X. Puig, C. Paxton, and *et al*., "Pre-trained language models for interactive decision-making," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 31199–31212, 2022.

[19] D. Driess, F. Xia, M. S. M. Sajjadi, and *et al*., "Palm-e: An embodied multimodal language model," *International Conference on Machine Learning (ICML)*, vol. 202, pp. 8469–8488, 2023.

[20] A. Zeng, M. Attarian, B. Ichter, and *et al*., "Socratic models: Composing zeroshot multimodal reasoning with language," *International Conference on Learning Representations (ICLR)*, 2023.

[21] L. Wang, Y. Zhang, and J. Feng, "On the Euclidean distance of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.27, no. 8, pp. 1334-1339, Aug. 2005.

[22] M. Likhachev, D. Ferguson, G. Gordon, and *et al*., "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613-1643, 2008.

[23] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1879-1884, Oct. 2009.

[24] J. D. Gammell, T. D. Barfoot and S. S. Srinivasa, "Batch informed trees(BIT): Informed asymptotically optimal anytime search," *International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, Apr. 2020.

[25] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 995–1001, Apr. 2000.

[26] L. E. Kavraki, P. Svestka, J.-C. Latombe, and *et al*., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[27] J. Funke, P. Theodosis, R. Hindiyeh, and *et al*., "Up to the limits: Autonomous Audi TTS," *IEEE Intelligent Vehicles Symposium*, pp. 541–547, Jun. 2012.

[28] W. Xu, J. Wei, J. M. Dolan, and *et al*., "A real-time motion planner with trajectory optimization for autonomous vehicles," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 51, pp. 2061–2067, Jun. 2012.

[29] D. Gonzlez, J. Prez, R. Lattarulo, and *et al*., "Continuous curvature planning with obstacle avoidance capabilities in urban scenarios," *IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1430–1435, Nov. 2014.

[30] O Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 2, pp. 84–90, Jun. 2012.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, and *et al*., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[33] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, Jun. 2019.

[34] M.-B. Radac and T. Lala, "Hierarchical cognitive control for unknown dynamic systems tracking," *Mathematics*, vol. 9, no. 21, p. 2752, Oct. 2021.

[35] E. Karpas and D. Magazzeni, "Automated planning for robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 417–439, 2020.

[36] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," *Proceedings of the 2020 International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 30, no.1, pp. 440–448, 2020.

[37] S. Liu, Z. Zeng, T. Ren, and *et al*., "Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection," *European Conference on Computer Vision*, Cham: Springer Nature Switzerland, pp. 38–55, 2024.

[38] A.D. Vuong, M.N. Vu, H. Le, and *et al*., "Grasp-anything: Large-scale grasp dataset from foundation models," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14030–14037, 2024.

[39] S. Hochreiter and J. Schmidhuber, " Long short-term memory," *Neural computation*, pp. 1735–1780, 1997.

[40] E.Coumans and Y.Bai, "Pybullet, a python modulefor physics simulation in robotics, games and machine learning,", 2017.

[41] T. Haarnoja, et al., "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." In Proc. International Conference on Machine Learning (ICML), 1861-1870 (2018).