Data-Agnostic Robotic Long-Horizon Manipulation with Vision-Language-Guided Closed-Loop Feedback

Yuan Meng[1], Xiangtong Yao[1], Haihui Ye[1], Yirui Zhou[1], Shengqiang Zhang[2], Zhenshan Bing[3,†], and Alois Knoll[1] *IEEE fellow*

***Abstract*— Recent advances in language-conditioned robotic manipulation have leveraged imitation and reinforcement learning to enable robots to execute tasks from human commands. However, these methods often suffer from limited generalization, adaptability, and the lack of large-scale specialized datasets, unlike data-rich domains such as computer vision, making long-horizon task execution challenging. To address these gaps, we introduce DAHLIA, a data-agnostic framework for language-conditioned long-horizon robotic manipulation, leveraging large language models (LLMs) for real-time task planning and execution. DAHLIA employs a dual-tunnel architecture, where an LLM-powered planner collaborates with co-planners to decompose tasks and generate executable plans, while a reporter LLM provides closed-loop feedback, enabling adaptive re-planning and ensuring task recovery from potential failures. Moreover, DAHLIA integrates chain-of-thought (CoT) in task reasoning and temporal abstraction for efficient action execution, enhancing traceability and robustness. Our framework demonstrates state-of-the-art performance across diverse long-horizon tasks, achieving strong generalization in both simulated and real-world scenarios. Videos and code are available at `https://ghiara.github.io/DAHLIA/`.**

I. INTRODUCTION

Language-conditioned robotic manipulation is an emerging field at the intersection of robotics, natural language processing, and computer vision, which aims to enable robots to interpret human commands and perform complex tasks using multi-modal sensing [1]. Imitation learning (IL) and reinforcement learning (RL) have traditionally been the dominant approaches for training robotic manipulation policies. However, recent IL and RL methods are often constrained to narrow task distributions, leading to sampling inefficiency and high sensitivity to distributional shifts, which limits their ability to generalize to diverse and complex scenarios. Additionally, both IL and RL are data-driven, requiring large-scale expert demonstrations, yet Internet-scale data collection for embodied AI remains a substantial challenge. In contrast, the natural language processing domain has seen state-of-the-art (SOTA) LLMs like GPT [2] and Llama [3] achieve human-like semantic understanding and common sense reasoning by training on massive datasets. Within embodied AI, LLMs offer a promising solution to bridge the gap between high-level language instructions and low-level robotic control, enabling robots to interpret abstract commands, perform spatial reasoning, decompose tasks into subgoals, and execute complex long-horizon tasks with improved adaptability.

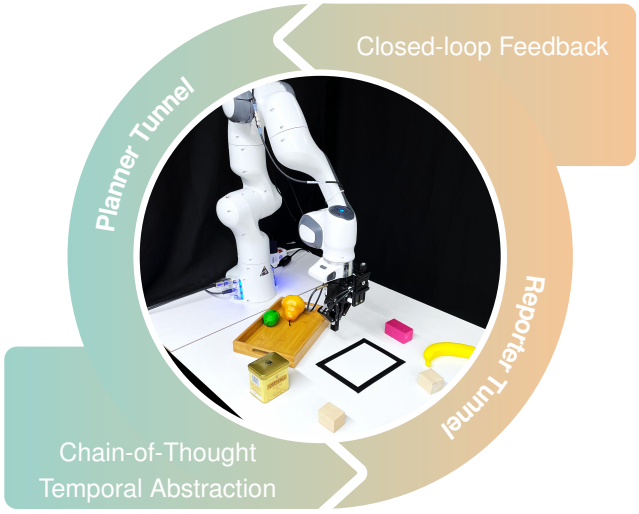


Fig. 1: Overview of our proposed framework. The dual-tunnel structure generates executable task plans and enables closed-loop feedback. By incorporating temporal abstraction and CoT-based guidance, our approach substantially enhances agent performance in both simulation and real-world long-horizon manipulation tasks.

In this study, we present an effective framework (Fig. 1) that leverages the common sense and reasoning capabilities of LLMs to enhance the flexibility and generalization of robotic long-horizon manipulation, eliminating the need for massive expert data. Departing from recent IL or RL approaches, our framework employs LLM as the task planner to directly interpret multimodal inputs and generate high-level task plans in deployable code format, invoking a library of motion primitives such as grasping and placing objects. To improve computational efficiency and inference robustness in long-horizon tasks, our framework integrates temporal abstraction [4], [5], enabling the agent to execute multiple steps before state reasoning, thereby reducing interference from per-step inference. Additionally, we incorporate chain-of-thought reasoning [6], [7], [8] in prompt engineering, making the model's reasoning process explicitly traceable and enhancing its task decomposition and decision-making capabilities. Upon task execution, another LLM serves as a reporter, evaluating the final state in a closed-loop manner. The reporter processes RGB-D images of the task scene along with textual prompts from the planner, returning task

[1]Yuan Meng, Xiangtong Yao, Haihui Ye, Yirui Zhou, and Alois Knoll are with the School of Computation, Information and Technology, Technical University of Munich, Germany.

[2]Shengqiang Zhang is with the Center for Information and Language Processing, Ludwig Maximilian University of Munich, Germany.

[3]Zhenshan Bing is with the State Key Laboratory for Novel Software Technology, Nanjing University, China.

[†] Corresponding author: `zhenshan.bing@tum.de`

completion judgments. This feedback informs the planner's subsequent decisions, ensuring dynamic re-planning to recover sub-task failure when necessary. This architecture enables real-time adaptive behavior, allowing the agent to effectively handle complex long-horizon manipulation tasks while demonstrating robust performance in unstructured and unseen environments. The key contributions of our study are summarized as follows: (1) We propose a data-agnostic framework that leverages a dual tunnel to bridge high-level language instructions and visual observations with low-level actions, enabling adaptive re-planning and robust long-horizon task execution without relying on massive expert datasets. (2) Our framework incorporates a temporal abstraction approach with the CoT technique, enabling interpretable reasoning and substantially reducing reasoning interference for long-horizon tasks. (3) Our framework demonstrates superior generalization and adaptability across diverse, long-horizon tasks in both simulation and real-world scenarios, achieving SOTA performance in both seen and unseen tasks. We name the framework as **DAHLIA**: **D**ata-**A**gnostic **H**ierarchical Framework for Language-guided **L**ong-horizon **I**nference and **A**ction Feedback.

II. RELATED WORK

Language-conditioned robotic manipulation combines robotics, natural language processing, and/or computer vision, using imitation learning or reinforcement learning to help robots follow human instructions [1], [9], [10]. RL methods rely on trial-and-error optimization guided by reward signals to teach robots task execution. By incorporating language embeddings into input representations, RL approaches can enhance the contextual understanding of tasks [11], [12], [13], [14], [15], [16], [17]. IL-based approaches, on the other hand, use offline datasets of human demonstrations combined with grounded language instructions to train agents that mimic observed action patterns [18], [19], [20], [21], [22], [23], [24]. However, both RL and IL approaches face challenges, including sample inefficiency or distributional shifts, limiting their adaptability to complex, long-horizon tasks, and generalization in unseen scenes.

Methods empowered by large-scale foundation models. Vision-language-action (VLA) models[25], [26], [27], [28], [29] represent a substantial advancement in embodied AI by integrating vision, language, and action modalities to enable robots to execute complex, multi-modal tasks. These models leverage pre-trained vision-language model backbones to directly generate actions, with either end-to-end or modulized structures [25]. Typical examples include RT-2 [30], RT-H [31], OpenVLA [32], π_0 [33]. However, to achieve robust generalization capabilities, VLA models typically require massive expert-level datasets and large-scale cluster computational resources. Efforts such as Open X-Embodiment [34] have introduced diverse datasets to improve model generalization across multiple tasks and embodiments. However, the scale of these datasets remains limited compared to the vast resources available in computer vision and natural language processing. This disparity in data scale continues to hinder the realization of human-like generalization capabilities for robotic systems in unstructured environments, leaving a gap for future research. In contrast, inspired by recent works [35], [36], [37], [38], our study advances this field by leveraging the common sense and world model knowledge embedded in LLMs [2]. By directly guiding robotic long-horizon manipulation tasks through a closed-loop framework, our method eliminates the need for extensive expert data required in VLA models, offering a more efficient and scalable solution.

III. METHOD

In this section, we present our proposed LLM/VLM-empowered closed-loop framework for embodied long-horizon manipulation tasks.

a) Preliminaries: A robotic manipulation task can be mathematically modeled as a Markov Decision Process (MDP), which is defined by a tuple (S,A,P,R). Here, S represents the state space of the environment while A denotes the action space available to the agent. The state transition function, $P : S \times A \to S$ governs how the environment transitions from the current state S_t to the next state S_{t+1} based on the action A_t, expressed as

$$P^a_{ss'} = P[S_{t+1} = s' | S_t = s, A_t = a]. \tag{1}$$

$R : S \times A \to \mathbb{R}$ is the reward function that maps the state S_t to reward R_{t+1} of next step according to action A_t such that

$$R^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]. \tag{2}$$

The overall objective is to find an optimal policy π^* so that the agent can perform the task trajectory with maximal accumulated expectation G:

$$G = \arg\max_{\pi^*} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \right], \tag{3}$$

where $\gamma \in [0,1]$ represents the discount factor, which weighs the importance of future rewards versus immediate rewards.

b) User Prompting: Since our framework discards learning-based policy models for action generation, the output of LLMs must be directly executable as code-plan-based primitives in the environment. To ensure LLMs understand their roles and generate the desired output, as shown in Fig. 2 (light green part on the left), we provide necessary prior knowledge through commented prompts before the task begins. This prior knowledge includes: (1) the available APIs, (2) the coordinate system, (3) general rules, and (4) example task plans for adaptation.

In our framework, low-level action patterns are implemented through Application Programming Interfaces (APIs), which define fundamental functionalities for robot actions and scene sensing. These APIs include acquiring object names, poses, and bounding boxes, detecting free spaces, and performing pick-and-place motions with either a two-claw gripper or a suction gripper. The detailed API instructions are provided in Table I. These APIs are introduced to the LLM alongside necessary third-party libraries, such as NumPy

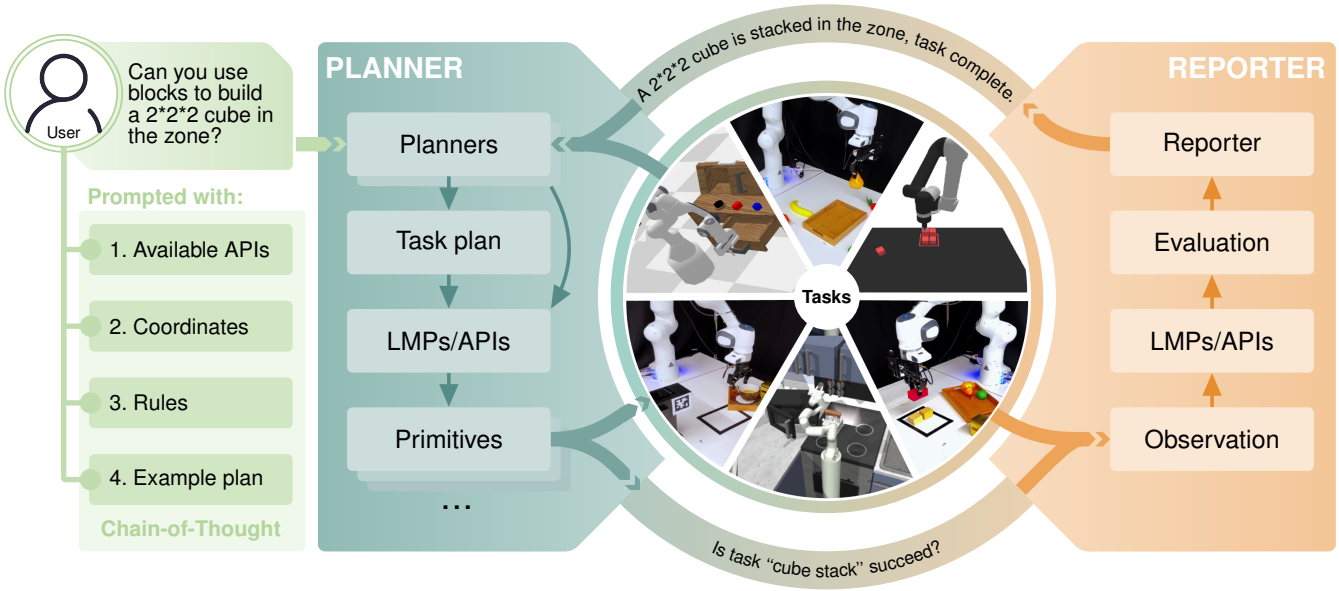


Fig. 2: Framework of DAHLIA. Our framework utilizes a dual-tunnel structure to implement a planner-reporter closed-loop feedback mechanism for various long-horizon manipulation tasks. The planner, powered by LLMs, converts high-level language instructions and initial state into task plans, leveraging available LMPs/APIs to generate multi-step motion primitives with the idea of temporal abstraction. The reporter, powered by a VLM, evaluates task outcomes using visual observations and provides feedback to the planner, enabling dynamic re-planning and robust performance in unstructured environments.

and itertools. The LLMs only need to understand their functionality, required arguments, formats, and the structure of their outputs.

TABLE I: Instructions of Basic APIs.

API Name	Instruction
get_obj_names()	Retrieves all objects in the environment with attributes such as color, shape, and unique IDs.
get_obj_pos(obj), get_obj_rot(obj)	Obtains the 3D position and rotation (Euler angle or quaternion) of a specific object (**obj**).
get_bbox(obj)	Provides the axis-aligned bounding box of an object (**obj**) to aid in spatial inference.
denormalize(pos)	Converts normalized coordinates (**pos**) into actual environment coordinates via linear transformation.
is_target_occupied(targ)	Checks if a target position or object (**targ**) is occupied and returns the names of occupying objects.
get_random_free_pos(targ, area)	Returns a random free position within a specified area (**area**) that does not occupy the target (**targ**).
put_first_on_second(obj1, obj2)	Executes a pick-and-place primitive to position (**obj1**) on (**obj2**) or a specified pose.

Notably, the agent can call the "denormalize(pos)" for complex spatial reasoning tasks, allowing it to work within its standard normalized coordinate system. However, in most cases, the environment coordinate system is implicit, and agents may not know the exact coordinates of a specific location. Instead, agents often use themselves as a reference and describe tasks with common directional terms from everyday language. Since such tasks typically involve simple comparisons along coordinate axes, we aim to avoid unnecessary coordinate transformations. Therefore, the second part of the prompt provides the LLMs with the scene's orientation and coordinate axes for direct comparisons, such as "front: x+", "left: y-", and "top: z+".

Additionally, prompts are delivered to the LLMs as natural language or key-value pairs embedded in Python comments. While some LLMs can process formatted prompts without additional instructions, detailed rules are included in natural language comments to ensure stable and accurate outputs. For instance, the following rule defines the role of a co-planner, which parses object information using APIs and assists the main planner:

> *"You are a task planning assistant who only answers with Python code. You are writing Python code for object parsing, refer to the code style in the examples below. You can use the existing APIs above, you must NOT import other packages."*

These rules inform the LLM about the current problem, direct it to refer to APIs and coordinate orientations, and guide task execution.

To further enhance the LLM's reliability for long-horizon reasoning, we include example task plans of increasing difficulty in the comments. These examples help the LLMs learn task patterns and generalize them to handle more complex scenarios. However, providing more examples alone is insufficient for LLMs to handle diverse, complex tasks, as the LLM's output becomes unreliable when faced with tasks dissimilar to the examples. To address this, we incorporate the CoT technique [6] in the examples, guiding the model to reason step-by-step rather than simply matching patterns. This approach enables the LLMs to not only handle tasks similar to the examples but also disassemble and analyze new tasks logically, producing more reliable and adaptable plans. For more details about prompting, refer to our code.

c) Dual-Tunnel Architecture: The framework detail is illustrated in Fig. 2, which consists of two main components to form a closed loop: the planner and the reporter. Inspired by Code-as-Policy [36], we observed that an LLM can

function directly as a policy based on state inputs. In the planner tunnel (dark green part), we designed a main planner combined with co-planners (powered by LLMs) to enable direct robot control by generating high-level task plans based on user commands. This approach not only enhances interpretability but also reduces the instability and noise associated with action token generation, as observed in previous works [30], [32]. With the underlying APIs, the main planner can directly access scene information without relying on inference from multimodal inputs. This allows it to focus on planning task scenarios using commonsense reasoning and world knowledge, substantially reducing cumulative errors and the computational burden associated with processing multi-modal information. Task descriptions provided by the user often include complex attributes to identify objects or manipulation targets, such as "the third block from the left that is different in color from the rightmost block but the same size". While the planner can utilize APIs to translate these descriptions into specific object names or coordinates, it is also responsible for formulating comprehensive task plans and arranging motion primitives. To address this challenge, our framework offloads detailed operations, such as handling objects and poses, to auxiliary LLMs, referred to as co-planners. By adapting the example task plans introduced by users, the co-planners can perform LLM-driven functions called language model programs (LMPs), which assist the main planner by decomposing complex information. When the main planner encounters a challenging problem, it delegates the task to the relevant LMP, which leverages the basic APIs to retrieve data and returns the processed information to the planner. The primary LMPs used in our framework are detailed in Table II.

TABLE II: Instructions of LMPs.

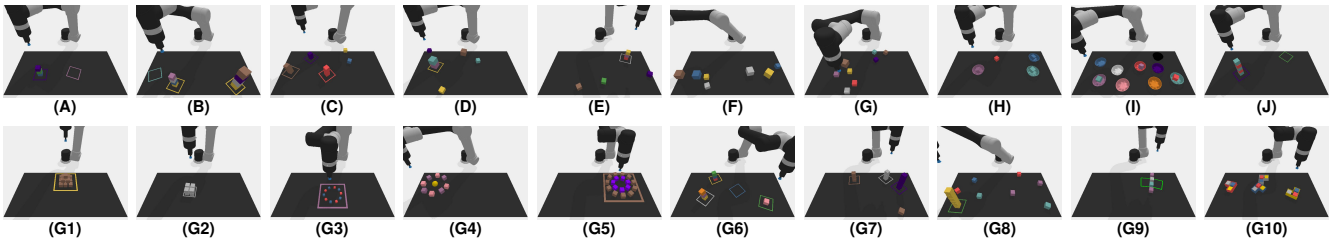
LMP Name	Instruction
parse_obj_name(dsc, ctxt)	Filters and returns object names from the context (**ctxt**) that match the target description (**dsc**) provided by the planner.
parse_position(dsc)	Converts a language description (**dsc**) of a location into one or more position coordinates or poses.
parse_function(dsc)	Implements new APIs by parsing the planner's description (**dsc**) and generating their functionality automatically.
parse_completion(dsc, ctxt)	Evaluates task completion from the context (**ctxt**) that match the target description (**dsc**).

Once the main planner generates a task plan, the configured LMPs and APIs are recurrently invoked by the environment to execute multiple robotic primitives. By incorporating temporal abstraction [4], [5], our framework enables multi-step action execution without requiring per-step planning and inference, unlike prior models [38]. This approach substantially reduces computational costs and minimizes error accumulation, enhancing the efficiency and reliability of long-horizon tasks. However, the planner is unaware of whether each primitive is executed successfully. To address this, we introduce a reporter tunnel to establish a closed-loop feedback mechanism. As shown in Fig. 2 (orange part), the reporter does not provide feedback for individual primitives but instead evaluates the overall environment status after all primitives are executed. The reporter takes the task goal description (dsc) and the scene observation (ctxt) as input, utilizing the LMP "parse_completion(dsc, ctxt)" to output a Boolean value indicating whether the task is completed. The input can include data extracted via LMPs/APIs or a pair of RGB images capturing the task's initial and final states from a fixed camera. If the reporter determines that the task has failed, the planner generates a revised task plan, re-executing the primitives. This process continues within a limited loop until the reporter confirms task completion.

IV. EXPERIMENT

a) Experiment Setup: To evaluate our framework's long-horizon manipulation performance and scalability across diverse scenarios, we utilize 22 long-horizon tasks from three tabletop manipulation benchmarks: LoHoRavens [38], CALVIN [39], and Franka Kitchen [40]. LoHoRavens, built on the Ravens simulator, focuses on language-conditioned tabletop manipulation using a UR5e robot arm equipped with a suction gripper. The environment features objects like blocks, bowls, and zones with variations in color, size, and texture, observed through RGB-D reconstructions and a top-down RGB view. As shown in Fig. 3, we expand the LoHoRavens task pool to 20 tasks using GenSim [41], including 10 tasks from the original pool (A-J) and 10 newly generated more complex tasks (G1-G10) for generalization evaluation. Ravens' matching matrix is used for evaluating original tasks, while generated tasks are manually assessed based on the final states, with any errors during execution considered as failures. CALVIN offers four indoor setups featuring a Franka Panda interacting with objects like drawers, sliding doors, buttons, and colored blocks. Similarly, the Franka Kitchen provides a kitchen scene where a Franka Panda manipulates appliances such as microwaves, burners, lights, and cabinets. A whole long-horizon task consists of four randomly selected subtasks. For real-world deployment, we evaluate our framework on a Franka robot equipped with an Intel RealSense D456 depth camera mounted at the table edge. To enable open-world object detection from inferred prompts, we use Grounded SAM2 (DINO-X + SAM2) [42]. For tasks requiring precise manipulation, such as pressing a button, we utilize AprilTags for accurate positioning.

LoHoRavens also provides VLM/LLM-empowered planner-reporter models for baseline comparison. The key differences between these baseline models and our framework are as follows: (1) Baseline models incorporate an additional CLIPort model [35] as a downstream action policy, fine-tuned for specific tasks, while our framework eliminates CLIPort entirely, using the LLM as a planner to directly generate executable primitives in the task plan. (2) Baseline models rely on per-step reasoning, making them more prone to noise and cumulative errors in action generation, which hinders performance in long-horizon tasks. In contrast, our framework leverages temporal abstraction combined with the CoT technique, substantially enhancing success rates in complex long-horizon scenarios. These baseline variations are thoroughly compared with our framework in the given tasks shown in Fig. 3.



Label	Instructions of Original Tasks	Label	Instructions of Generated Tasks
A	"Stack all blocks in the [COLOR] zone."	G1	"Construct a 9-4-1 rectangular pyramid structure in the zone using 14 blocks of the same color."
B	"Stack blocks of the same size in the [COLOR1] zone and [COLOR2] zone respectively."	G2	"Construct a 2*2*2 cube structure in the zone using 8 blocks of the same color."
C	"Stack all the blocks of the same color together in the same colored zone."	G3	"Construct a circle with suitable radius with alternating [COLOR1] and [COLOR2] blocks in the zone."
D	"Stack only the [SIZE] blocks of [COLOR_TYPE] color in the [COLOR] zone."	G4	"Construct a circle with suitable radius with alternating [COLOR1] and [COLOR2] blocks around the ball."
E	"Stack all the blocks, which are to the [REL_POS] of the [COLOR1] block with [POS_TYPE] distance larger than 0.05 unit, in the [COLOR2] zone."	G5	"Construct two concentric circles in the zone using [NUM] [COLOR1] and [NUM + 4] [COLOR2] blocks."
F	"Move all the blocks in the [POS1] area to [POS2] area."	G6	"Divide the blocks into groups of [NUM] and stack each group (also including the group with block number less than [NUM]) in a different zone."
G	"Move all the [SIZE] blocks in the [POS1] area to [POS2] area."	G7	"Place the maximal odd number of blocks of the same color in each correspondingly colored zone."
H	"Put the blocks in the bowls with matching colors."	G8	"Stack blocks of the same color that has the largest quantity in the zone."
I	"Put the blocks in the bowls with mismatching colors."	G9	"Arrange all blocks on the zone bisector line between two symmetrically placed zones evenly on the tabletop, and the gap between two adjacent blocks' edges should be near the block size, and the line connecting the center of the zones also bisects these blocks."
J	"Stack blocks with alternate colors on the [COLOR1] zone, starting with the [COLOR2] color."	G10	"Each L-shaped fixture can hold three blocks, suppose the block size is (a,a,a), then in fixture's local coordinate system, the three places that can hold blocks are [(0,0,0),(a,0,0),(0,a,0)]. Fill in all the fixtures which have random position and rotation with blocks, and make sure in the end in every fixture there are three blocks with different colors."

Fig. 3: Completion snapshots of LoHoRavens Benchmark and generated tasks. The first row shows tasks from LoHoRavens, and the second row demonstrates generated tasks. The table shows the corresponding task instructions.

In this work, we use OpenAI GPT-4o-mini [2] as the backbone for both the planner and reporter of our default framework, handling all task planning, reporting, and generation. Additionally, we perform ablation studies to assess the impact of the dual-tunnel structure with closed-loop feedback, where we evaluate a planner-only variant that generates and performs task plans without reporter feedback. The results demonstrate that our default framework setup achieves the best performance compared to this variation. All experiments are conducted using up to a maximum of 8 Nvidia V100 GPUs (32GB) for distributed inference. Each task is conducted with 50 random seeds, and the average success rate (%) is reported for comparison.

b) Long-horizon Manipulation: First, we evaluate the performance of all frameworks on the original LoHoRavens task pool. As shown in Table III, our proposed framework, DAHLIA, outperforms all baseline models in long-horizon manipulation tasks, achieving the highest success rates, including 100% accuracy in five out of ten tasks. This demonstrates the effectiveness of hierarchical task planning and closed-loop feedback, enabling adaptive execution and error correction. Compared to its planner-only variant (DAHLIA GPT-4o-mini^{P}), our dual-tunnel framework provides a clear advantage, particularly in tasks requiring precise coordination and iterative re-planning. The success rate improvement in certain tasks reaches up to 12%, highlighting the benefit of iterative evaluation and adaptive re-planning. The dual-tunnel system improves reliability by allowing the reporter to assess the task state after each planning loop, detect failures, and trigger re-planning to recover failed subtasks in subsequent loops, leading to more stable and efficient execution in long-horizon tasks. In contrast, baseline models relying on per-step inference and CLIPort for action execution exhibit significantly lower performance. While GPT-4o-mini^{P} + GPT-4o-mini^{R} achieves relatively higher success rates than other baselines, its dependence on per-step inference causes cumulative errors, making it susceptible to execution noise, with an average success rate below 32% across all tasks. Additionally, since all baselines rely on CLIPort, their generalization capability is constrained even after fine-tuning on task-specific data. By eliminating per-step inference and instead using temporal abstraction and structured task planning, DAHLIA effectively reduces error accumulation and ensures stable task execution.

TABLE III: Success rates for individual tasks from the original LoHoRavens task pool.

	Long-horizon Task Average Success Rate (%)									
Frameworks	A	B	C	D	E	F	G	H	I	J
*CLIPort (oracle)	22	2	8	2	2	16	10	20	18	2
*Llama2^{P} + Flamingo^{R}	20	10	4	10	16	28	32	32	28	14
*Llama3^{P} + CogVLM2^{R}	20	20	24	22	14	20	20	32	30	22
*GPT-4o-mini^{P} + CogVLM2^{R}	26	22	30	24	20	34	28	36	36	30
*GPT-4o-mini^{P} + GPT-4o-mini^{R}	30	32	32	30	18	34	30	46	36	32
†DAHLIA (GPT-4o-mini^{P})	100	88	96	60	42	76	80	90	90	82
DAHLIA (Ours)	100	100	100	48	42	88	72	100	90	80

*Baseline combinations are adopted from the LoHoRavens [38], where all baselines use planner for per-step inference, reporter for explicit feedback and CLIPort for action execution. In contrast, our framework employs temporal abstraction to minimize noise from per-step inference and utilizes task plans for action execution instead of CLIPort, which may produce inaccurate action patterns. ^{P} indicates the planner model. ^{R} indicates the reporter model. † indicates the framework barely uses a planner-only tunnel without the reporter, resulting in open-loop control.

To further assess the framework's performance in more generalized and complex scenarios, we evaluate its execution on the newly generated tasks (G1-G10) shown in Fig. 3. The results in Table IV demonstrate that DAHLIA consistently achieves the highest success rates, further validating its robust generalization and stability in challenging scenarios. Our

framework achieves 100% success in five tasks and maintains high performance across most others, outperforming all baselines. Compared to the planner-only variant (DAHLIA GPT-4o-mini^P), incorporating closed-loop feedback leads to substantial performance improvements, particularly in G2 (+24%), G3 (+8%), G5 (+8%), and G7 (+12%), emphasizing the importance of iterative evaluation and re-planning. In contrast, baseline models exhibit substantially lower performance, with CLIPort (oracle) failing entirely on most tasks and the GPT-4o-mini^P + GPT-4o-mini^R combination, which performs better than other baselines in the original task pool, struggling with execution stability in such challenging scenarios. The primary limitations of the baselines stem from their reliance on per-step inference and CLIPort as the downstream action policy, which introduces cumulative errors and lacks adaptability in unseen complex tasks. Per-step inference generates actions sequentially without a global understanding of the task, making it prone to execution noise. Additionally, CLIPort, which was not fine-tuned for these tasks, struggles to generalize effectively, further degrading performance. In contrast, DAHLIA eliminates these issues by leveraging temporal abstraction and structured CoT-inspired task plans, ensuring execution stability while efficiently adapting to new tasks. The reporter further enhances robustness by enabling closed-loop error recovery and iterative refinement, leading to higher long-horizon task completion rates. These results confirm that DAHLIA's task planning and feedback-driven execution provide an alternative, scalable, and data-agnostic solution for complex robotic manipulation.

TABLE IV: Success rates for individual generated complex tasks.

	Long-horizon Task Average Success Rate (%)									
Frameworks	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
*CLIPort (oracle)	0	32	4	0	0	0	0	8	0	0
*GPT-4o-mini^P + GPT-4o-mini^R	8	36	34	30	4	28	30	18	0	0
†DAHLIA (GPT-4o-mini^P)	58	76	92	100	42	98	88	88	12	8
DAHLIA (Ours)	50	100	100	100	50	100	100	90	18	10

*Baseline combinations are adopted from the LoHoRavens [38], where all baselines use planner for per-step inference, reporter for explicit feedback and CLIPort for action execution. ^{P} indicates the planner model. ^{R} indicates the reporter model. † indicates the framework barely uses a planner-only tunnel without the reporter, resulting in open-loop control.

c) Scalability: To assess the scalability and generalization of our framework across different embodiments and task setups, we conduct additional experiments using the CALVIN and Franka Kitchen benchmarks. The average success rates for individual subtasks are presented in Table V. The results show that our default dual-tunnel DAHLIA setup consistently outperforms its planner-only variation (DAHLIA GPT-4o-mini^P), demonstrating the effectiveness of its closed-loop feedback mechanism in improving execution reliability. Our framework achieves 100% success in most tasks, including lift block, open drawer, open slide door, kettle, and slider cabinet, while also surpassing the planner-only model in complex tasks requiring precise coordination, such as rotate block (+4%) and turn on switch (+4%) in CALVIN, as well as top burner (+6%) and bottom burner (+8%) in Franka Kitchen. Notably, the planner-only model may struggle in tasks requiring highly accurate task planning and fine-grained manipulation motions. For example, in the "top burner" and "bottom burner" tasks in Franka Kitchen, the robot must accurately reach the correct rotary knob and rotate it counterclockwise by 45 degrees to activate the corresponding burner. Errors in task planning or dexterous execution—such as selecting the wrong switch or rotating in the wrong direction—can lead to complete task failure. In such cases, closed-loop feedback in the dual-tunnel plays a critical role by detecting and correcting execution errors, enabling the planner to refine its actions and complete the global task plan. The performance gap between the two DAHLIA variations is relatively small, as both employ temporal abstraction and CoT-based task planning, ensuring efficient long-horizon primitive execution. However, the closed-loop feedback mechanism can enhance robustness, allowing for error correction and ensuring greater stability in complex environments. These results further validate DAHLIA's robust execution and adaptive reasoning capabilities, demonstrating its ability to generalize across diverse tasks and embodiments.

TABLE V: Average success rates (%) for individual subtasks in CALVIN and Franka Kitchen

CALVIN Tasks	†DAHLIA (GPT-4o-mini^P)	DAHLIA (Ours)
lift block	100	100
rotate block	94	98
turn on switch	96	100
open slide door	100	100
open drawer	100	100
overall	98.00	99.60
Franka Kitchen Tasks	†DAHLIA (GPT-4o-mini^P)	DAHLIA (Ours)
microwave	98	100
kettle	100	100
light	98	100
top burner	90	96
bottom burner	90	98
slider cabinet	100	100
hinge cabinet	96	98
overall	96.00	98.86

^{P} indicates the Planner model. † indicates the framework barely uses a Planner-only tunnel for a open-loop control without the Reporter

The completion state snapshots in Fig. 4 showcase the effectiveness of our framework across diverse manipulation tasks, from structured tabletop interactions in CALVIN to articulated object handling in Franka Kitchen. These results validate DAHLIA's hierarchical task planning and feedback-driven execution, demonstrating its ability to generalize across new tasks and embodiments, making it a scalable and adaptable solution for long-horizon robotic manipulation.

d) Real-world Performance: The real-world zero-shot performance of our proposed framework is illustrated in Fig. 5. We evaluate our system on three long-horizon tabletop manipulation tasks: picking all fruits and placing them on a plate (first row), stacking blocks into a pyramid (second row), and making a coffee (third row). To increase the complexity

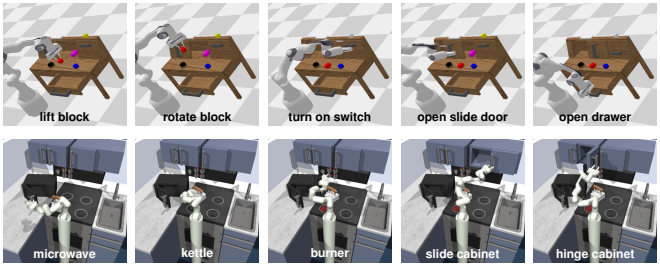


Fig. 4: Completion state snapshots of CALVIN and Franka Kitchen benchmarks. The first row shows illustrations from CALVIN, and the second row demonstrates images from Franka Kitchen.

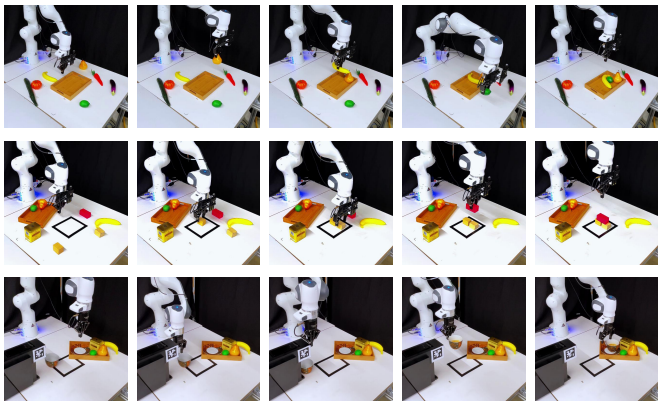


Fig. 5: Snapshots of real-world demonstrations. Pick all fruits and place them on the plate (1st row). Stack blocks in a pyramid shape (2nd row). Make the coffee from the coffee machine (3rd row).

of the framework's scene reasoning, we introduced additional objects into the task scenarios. For instance, in the first scenario, we added vegetable models such as cucumbers and pumpkins, requiring the planner to accurately identify and avoid incorrect selections. In the second scenario, we placed a rectangular tea can, challenging the agent to distinguish it from the wooden blocks used for construction. In the coffee-making task, dexterous operations such as pressing the coffee button require high-precision scene detection. However, due to the approximate 10mm depth measurement error of a single RealSense camera, accurately determining the exact position of the button is challenging. To ensure successful task execution, we calibrate the button's position using an AprilTag, providing the necessary precision for reliable manipulation. Before execution, the agent calls corresponding APIs to scan the scene based on the state-of-the-art open-world detection model, extracts object attributes (e.g., ID, bounding box, position and orientation etc.) and target information based on user instructions, and utilizes LMPs to generate executable task plans. It then executes the planned primitives, and at the end of the loop, the reporter evaluates the scene and determines the task completion status. As shown in the snapshots, our framework successfully completes all tasks in a single loop, demonstrating robust performance and real-world generalization. Full video renderings are available on our project website[1].

V. CONCLUSIONS

We present DAHLIA, a data-agnostic hierarchical framework for long-horizon robotic manipulation, leveraging LLMs/VLMs with a dual-tunnel closed-loop feedback mechanism to improve execution stability and generalization. By integrating temporal abstraction in primitive execution and CoT-based reasoning in task planning, DAHLIA eliminates the need for extensive task-specific training while enhancing adaptability and efficiency. Evaluations on various long-horizon manipulation task benchmarks, including LoHo-Ravens, CALVIN, and Franka Kitchen, as well as newly generated long-horizon tasks using Gensim, demonstrate state-of-the-art performance over baselines, particularly in scenarios requiring error correction and iterative re-planning. Future work will focus on adapting DAHLIA to temporally non-stationary environments, improving spatial reasoning for precise manipulation, and extending to multi-agent collaboration. By providing a scalable, data-efficient approach, DAHLIA advances language-conditioned robotic manipulation toward more adaptive and generalizable embodied AI systems.

[1]Videos can be accessed at `https://ghiara.github.io/DAHLIA/`.

ACKNOWLEDGMENT

The authors acknowledge the financial support by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) for the Lighthouse Initiative KI.FABRIK (Phase 1: Infrastructure as well as the research and development program under grant no. DIK0249).

REFERENCES

[1] H. Zhou, X. Yao, Y. Meng, S. Sun, Z. BIng, K. Huang, and A. Knoll, "Language-conditioned learning for robotic manipulation: A survey," *arXiv preprint arXiv:2312.10807*, 2023.

[2] 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.

[3] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[4] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[5] K. Pertsch, Y. Lee, and J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conference on robot learning*. PMLR, 2021, pp. 188–204.

[6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[7] Z. Yu, L. He, Z. Wu, X. Dai, and J. Chen, "Towards better chain-of-thought prompting strategies: A survey," *arXiv preprint arXiv:2310.04959*, 2023.

[8] X. Wang and D. Zhou, "Chain-of-thought reasoning without prompting," *arXiv preprint arXiv:2402.10200*, 2024.

[9] R. Miao, Q. Jia, and F. Sun, "Long-term robot manipulation task planning with scene graph and semantic knowledge," *Robotic Intelligence and Automation*, vol. 43, no. 1, pp. 12–22, 2023.

[10] G. Tejus, G. Zara, P. Rota, A. Fusiello, E. Ricci, and F. Arrigoni, "Rotation synchronization via deep matrix factorization," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2113–2119.

[11] S. Sodhani, A. Zhang, and J. Pineau, "Multi-task reinforcement learning with context-based representations," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9767–9779.

[12] A. Silva, N. Moorman, W. Silva, Z. Zaidi, N. Gopalan, and M. Gombolay, "Lancon-learn: Learning with language to enable generalization in multi-task manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1635–1642, 2021.

[13] P. Goyal, S. Niekum, and R. Mooney, "Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards," in *Conference on Robot Learning*. PMLR, 2021, pp. 485–497.

[14] Y. Meng, Z. Bing, X. Yao, K. Chen, K. Huang, Y. Gao, F. Sun, and A. Knoll, "Preserving and combining knowledge in robotic lifelong reinforcement learning," *Nature Machine Intelligence*, pp. 1–14, 2025.

[15] D. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1004–1015.

[16] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[17] R. Kaplan, C. Sauer, and A. Sosa, "Beating atari with natural language guided reinforcement learning," *arXiv preprint arXiv:1704.05539*, 2017.

[18] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn *et al.*, "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation," in *Conference on Robot Learning*. PMLR, 2022, pp. 1303–1315.

[19] H. Zhou, Z. Bing, X. Yao, X. Su, C. Yang, K. Huang, and A. Knoll, "Language-conditioned imitation learning with base skill priors under unstructured data," *IEEE Robotics and Automation Letters*, 2024.

[20] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning over unstructured data," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 205–11 212, 2022.

[21] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[22] Z. Xian and N. Gkanatsios, "Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation," in *Conference on Robot Learning/Proceedings of Machine Learning Research*. Proceedings of Machine Learning Research, 2023.

[23] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *arXiv preprint arXiv:2304.02532*, 2023.

[24] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Conference on Robot Learning*. PMLR, 2023, pp. 3766–3777.

[25] Y. Ma, Z. Song, Y. Zhuang, J. Hao, and I. King, "A survey on vision-language-action models for embodied ai," *arXiv preprint arXiv:2405.14093*, 2024.

[26] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan, "3d-vla: A 3d vision-language-action generative world model," *arXiv preprint arXiv:2403.09631*, 2024.

[27] K. F. Gbagbe, M. A. Cabrera, A. Alabbas, O. Alyunes, A. Lykov, and D. Tsetserukou, "Bi-vla: Vision-language-action model-based system for bimanual robotic dexterous manipulations," in *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2024, pp. 2864–2869.

[28] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng *et al.*, "Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation," *arXiv preprint arXiv:2409.12514*, 2024.

[29] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[30] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[31] S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi, and D. Sadigh, "Rt-h: Action hierarchies using language," *arXiv preprint arXiv:2403.01823*, 2024.

[32] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[33] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[34] A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[35] 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.

[36] 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.

[37] L. Zhang, K. Cai, Z. Sun, Z. Bing, C. Wang, L. Figueredo, S. Haddadin, and A. Knoll, "Motion planning for robotics: A review for sampling-based planners," *Biomimetic Intelligence and Robotics*, vol. 5, no. 1, p. 100207, 2025.

[38] S. Zhang, P. Wicke, L. K. Şenel, L. Figueredo, A. Naceri, S. Haddadin, B. Plank, and H. Schütze, "Lohoravens: A long-horizon language-conditioned benchmark for robotic tabletop manipulation," *arXiv preprint arXiv:2310.12020*, 2023.

[39] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022.

[40] A. Gupta *et al.*, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1025–1037.

[41] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang, "Gensim: Generating robotic simulation tasks via large language models," *arXiv preprint arXiv:2310.01361*, 2023.

[42] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.