Integrating Advantage Actor-Critic in Multi-Robot Collaboration

Jiazhao Liang†, Hao Huang†*, Yu Hao, Geeta Chandra Raju Bethala,
Congcong Wen, Shuaihang Yuan, Anthony Tzes, and Yi Fang*

***Abstract*—Recent advances in large language models (LLMs) have spurred interest in using these models to coordinate multi-agent robot systems. However, existing approaches often fail to handle dynamic and complex environments effectively. We present A2C-Collab, an advantage actor critic framework tailored to multi-robot collaboration. A2C-Collab contains three major components: (1) an actor generates time-critical commands for each robot to execute, and (2) a critic monitors execution and recommends corrections when plans fail. (3) An advantage mechanism verifies these corrections by forecasting their impact on subsequent environmental dynamics. While previous methods primarily relied on the critic to enhance collaboration, they often lacked a verification mechanism, allowing the critic to unintentionally guide agents away from the correct goal. In contrast, our approach introduces an advantage verification stage that anticipates and evaluates the impact of corrective actions before execution, ensuring more reliable and goal-aligned coordination. The framework was first evaluated in RoCoBench, a standard multi-robot simulation, and subsequently deployed to a physical robot cluster. Across both settings, A2C-Collab improved task completion rates compared with the state-of-the-art baselines, demonstrating robust performance and highlighting the promise of LLM-driven reasoning in real-world multi-robot systems. Code is available at: https://github.com/A2C-collab/A2C_collab.**

***Index Terms*—multi-robot systems, large language models, advantage actor-critic**

I. Introduction

MULTI-ROBOT systems are increasingly valuable for complex, long-horizon tasks in environments unsafe for humans [1]. However, they face two key challenges: reliable operation in dynamic environments and seamless coordination among robots [2], [3]. Traditional approaches emphasize low-level control while leaving high-level collaboration underdeveloped. Large language models (LLMs) address this by reasoning over unstructured information, interpreting high-level goals, and generating coordinated plans that adapt to environmental changes, improving multi-robot coordination and flexibility. However, LLM outputs can drift with long context or rapid changes, causing hallucinations or inconsistent priorities [4]. In multi-robot settings, such errors cascade and disrupt coordination. Current systems lack mechanisms to detect and correct these failures during execution.

Previous work [5] introduced a retrospective actor-critic framework for multi-robot teams but failed to verify critic corrections aligned with task objectives, causing suboptimal behaviors in complex scenarios. We introduce advantage actor-critic collaboration (A2C-Collab), a substantial enhancement that incorporates a verification mechanism to validate corrections before execution. Adapting retrospective critic and advantage estimation [6], we design a three-tier planner: an Actor LLM proposes actions from goals and context; a Critic LLM diagnoses faults and provides corrections; and an Advantage Verifier LLM evaluates whether corrections improve the plan. We validate through RoCo bench [7] simulations and real-world deployments. Our contributions are as follows:

- We introduce A2C-Collab, an LLM-based multi-robot framework that integrates advantage actor-critic (A2C) for complex tasks. The framework generates coordinated actions, uses critic to diagnose and correct failures, and employs advantage estimation to verify that corrections improve performance.
- A2C-Collab uses structured prompting and dual memory to coordinate actor, critic, and verifier LLMs, enabling efficient multi-step reasoning without overwhelming the language models.
- Simulated and real-world experiments validate the framework's effectiveness, demonstrating that A2C-Collab enhances collaboration accuracy and task success rates in challenging environments.

II. Related Work

Multi-robot Systems. Recent multi-robot research spans collision-free motion planning, embodied learning-based coordination, and cooperative perception. In motion planning, Lin *et al*. [8] propose distributional reinforcement learning for decentralized navigation, while Tajbakhsh *et al*. [9] and Moldagalieva *et al*. [10] combine conflict resolution with dynamic control for scalable coordination. For embodied learning, Fernando *et al*. [11] develop graph-attention strategies for UAV fleets, Liu *et al*. [12] design TraCo for traffic control, and Fawcett *et al*. [13] implement predictive control for legged robots. In adaptive coordination, Tzes *et al*. [14] use graph neural networks for decentralized sensing, Xu *et al*. [15] apply bandit-based optimization for decision-making, and Singh *et al*. [16] propose learning task rewards while forming adaptive coalitions and Bezerra *et al*. [17] develop learning-based policies for dynamic coalition formation under partial observability. Unlike previous works, this study presents an advantage actor-critic formulation for language-grounded multi-robot collaboration. While the A2C framework has been applied to domains like traffic signal optimization [18], our framework focuses on collaborative manipulation, enforcing symbolic action generation and verifying corrective actions to resolve execution conflicts.

†Equal contribution to this work.

Jiazhao Liang is with New York University Tandon School of Engineering, Brooklyn, NY, USA.

Hao Huang, Yu Hao, Geeta Chandra Raju Bethala, Congcong Wen, Shuaihang Yuan, Anthony Tzes, and Yi Fang are with NYUAD Center for Artificial Intelligence and Robotics (CAIR) and Embodied AI and Robotics (AIR) Lab, New York University Abu Dhabi, UAE.

Corresponding authors: Hao Huang (hh1811@nyu.edu) and Yi Fang (yfang@nyu.edu).

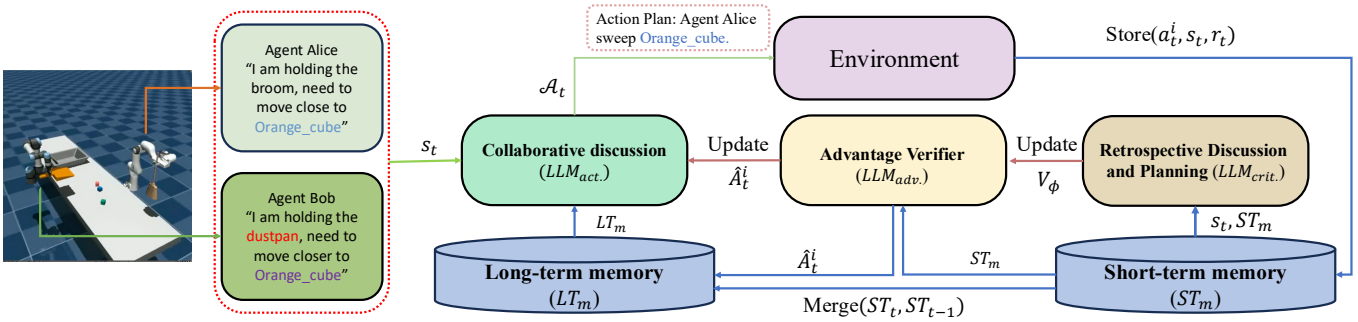

Fig. 1. A2C-Collab coordinates two robotic arms (Alice and Bob) using three LLMs: $LLM_{act.}$, $LLM_{crit.}$, and $LLM_{adv.}$. The process starts with collaborative discussion where agents propose action plans and assign tasks. After validating actions through inverse kinematics and collision checks, the plan is stored in short-term memory ST_m and sent to $LLM_{crit.}$ for critic. $LLM_{crit.}$ proposes revisions, which $LLM_{adv.}$ verifies for improvement. Verified information is stored in long-term memory LT_m, retaining only the two most recent rounds as input for $LLM_{act.}$'s next planning cycle.

LLM in Robotics. Recent research leverages LLMs for robot manipulation. Brohan *et al.* [19] propose SayCan, filtering LLM actions by affordance scores, while Liang *et al.* [20] introduce Code-as-Policies to generate robot control programs. Singh *et al.* [21] extend this with ProgPrompt, constraining plans to known skills for reliable execution. Rana *et al.* [22] develop SayPlan using 3D scene graphs for spatial reasoning, and Joublin *et al.* [23] propose CoPAL with corrective feedback to improve manipulation accuracy. Zitkovich *et al.* [24] advance this with RT-2, enabling zero-shot generalization through vision-language-action models. Mandi *et al.* [7] scale to multi-robot manipulation with RoCo, combining high-level language goals with distributed planning. Building on these foundations, we propose A2C-Collab that enables multi-robot teams to refine LLM-guided manipulation by learning from execution failures.

III. Method

A. Overview

The proposed A2C-Collab framework, shown in Figure 1, comprises three dedicated LLMs and a dual-memory structure. The actor LLM ($LLM_{act.}$) generates real-time action commands from the current state, the critic LLM ($LLM_{crit.}$) evaluates execution outcomes, and the verifier LLM ($LLM_{adv.}$) generates the expected advantage of corrected actions to refine future decisions. Inspired by the advantage actor-critic paradigm [6], our framework separates actor, critic, and verifier into distinct LLM models to prevent cross-interference and ensure each model remains dedicated to its specific function. To maintain context awareness and facilitate corrective learning, we integrate a dual-memory structure in which short-term memory (ST_m) captures the current dialog history and long-term memory (LT_m) retains the two most recent interaction rounds as supplementary context [25]. Our A2C-Collab framework adopts a centralized decision-making architecture, where a single actor $LLM_{act.}$ generates coordinated actions for all N robots. We do not adopt decentralized strategies, as they require separate LLM instances per agent, causing memory and computational costs to grow proportionally with the number of robots.

At the start of each round, each robot receives an updated state s_t, its assigned role, and the joint goal g. The robots then engage in dialog to propose actions, which the actor model jointly generates for all robots at planning cycle t.

We define *consensus* as convergence to a feasible plan, achieved when $LLM_{act.}$ outputs a plan marked **EXECUTE** and all actions satisfy inverse-kinematics constraints. The consensus condition is written as:

$$\text{Consensus}(\mathcal{A}_t) = \mathbf{1}\{\textbf{EXECUTE}\} \wedge \bigwedge_{i=1}^{N} \text{IK\_validate}(a_t^i). \tag{1}$$

When consensus is reached on a proposal, it is recorded as the collective action $\mathcal{A}_t$ for all robots at time step t, written to short-term memory ST_m, and the consolidated plan is parsed into a sequence of Cartesian waypoints. $\{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$ specifying end-effector positions, paired with gripper actions (`PICK`, `PLACE`, `MOVE`, `SWEEP`, `WAIT`). Each waypoint is converted via inverse kinematics into joint configurations $q_t^i \in \mathbb{R}^{d_i}$ and refined using RRT [26] to ensure collision-free execution before deployment. After all robots commit their actions, the dialog ends with EXECUTE and the plan is parsed into low-level commands. The critic LLM ($LLM_{crit.}$) analyzes environment feedback to formulate corrections c_i, while the verifier LLM ($LLM_{adv.}$) validates these corrections and predicts the resulting state $\hat{A}_i^t$. The dialogs, actions, feedback, and verified revisions are stored in long-term memory (LT_m) to inform future planning rounds.

B. Problem Formulation

We formulate multi-robot collaborative manipulation as a language-grounded MDP defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote state and action spaces, $\mathcal{P}(s'|s,a)$ represents the transition probability to state s' given state s and action a, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0,1)$ is the discount factor. At timestep t, the state $s_t = \{o_t, g, \{\text{role}_i\}_{i=1}^N, \mathcal{H}_{t-2:t}\}$ consists of: (1) observation o_t containing robot joint positions $q_t^i \in \mathbb{R}^{d_i}$, gripper states $u_t^i \in \{0,1\}$, M object poses $\{p_t^j, \theta_t^j\}_{j=1}^M$ where $p_t^j \in \mathbb{R}^3$ and $\theta_t^j \in SO(3)$; (2) shared task goal g in natural language; (3) robot role assignments $\{\text{role}_i\}_{i=1}^N$; and (4) dialog history $\mathcal{H}_{t-2:t}$ from the two most recent interaction rounds. Each robot i generates a discrete symbolic action $a_t^i =$

$(\text{action\_type}^i, \text{target\_object}^i)$ through LLM-based dialog, where $\text{action\_type}^i \in \{\texttt{MOVE}, \texttt{PICK}, \texttt{PLACE}, \texttt{SWEEP}, \texttt{WAIT}\}$ and target_object^i target object or location. The complete action is $\mathcal{A}_t = \{a_t^1, \ldots, a_t^N\}$, generated as $a_t^i = \textit{LLM}_{act.}(s_t, P_t)$ where $P_t = \texttt{Construct}(s_t, g, LT_m)$ retrieves context from long-term memory. Before execution, actions undergo inverse kinematics validation $\text{IK\_validate}(a_t^i) \in \{\texttt{VALID}, \texttt{INVALID}\}$, and validated actions are executed to yield next state s_{t+1} and natural-language feedback r_t.

C. Prompt-driven Advantage Actor-critic Coordination

Our adaptation of A2C [6] extends the single-agent setting to N robots that coordinate via language-based negotiation. At each cycle, the environment provides a global state s_t encoding object poses, kinematic limits, and per-robot roles (*e.g.*, *Alice*, *Bob*, *Chad*). Given (s_t, g), the actor $LLM_{act.}$ with parameters φ assigns each robot $i \in \{1, \ldots, N\}$ a sub-goal g_t^i and a high-level action a_t^i at time $t \in \{0, 1, 2, \ldots\}$.

At each step, a robot-specific prompt P_t^i is retrieved from long-term memory LT_m, which stores the two most recent dialog rounds. The action at time t is generated as:

$$a_t^i = \textit{LLM}_{act.}(s_t, P_t^i), \qquad i = 1, \ldots, N. \tag{2}$$

The retrieved prompt provides historical grounding from LT_m, enabling $LLM_{act.}$ to produce actions consistent with the ongoing multi-robot dialog. Here, an action a_t^i represents the complete transformation from high-level language to low-level control. The $LLM_{act.}$ outputs structured text (*e.g.*, "Alice: PICK apple_top"), parsed into $e_t^i \in SE(3)$ and mapped via inverse kinematics to $q_t^i \in \mathbb{R}^{d_i}$. A collision-free trajectory is planned via RRT [26], downsampled to 15 Hz, and executed as position-controlled commands in MuJoCo [27]. For grasping, the actions specify a binary gripper state (open or closed).

An inverse-kinematics validator $\text{IK\_validate}(a_t^i)$ checks each action a_t^i at time t for robot i by verifying the existence of a joint configuration $q_t^i \in \mathbb{R}^{d_i}$ that realizes the target pose $e_t^i \in SE(3)$ via f_{FK} and lies within the space $\mathcal{Q}_{\text{free}}$. Formally:

$$\text{IK\_validate}(a_t^i) \triangleq \mathbf{1}\left\{\exists q_t^i : (f_{\text{FK}}(q_t^i) = e_t^i) \wedge (q_t^i \in \mathcal{Q}_{\text{free}})\right\} \tag{3}$$

This yields each command a feasibility label of VALID if the condition holds or INVALID otherwise, rejecting actions with IK failure, unreachable poses, joint limit violations, or collisions. Each round is written to short-term memory ST_m. When all robots propose feasible commands, the dialog ends with EXECUTE and action a_t^i is executed, yielding the next state s_{t+1} and reward r_t. Although environment feedback is in natural language (*e.g.* "collision detected; replan required"), we map it to scalar rewards: $r_t = 1$ (success), $r_t = -0.5$ (partial success), or $r_t = -1$ (failure). Partial success occurs when one component succeeds but another fails, such as valid IK with collision during execution, or collision-free trajectory with infeasible IK solution. This asymmetric reward design encourages successful actions while providing intermediate feedback for partial success.

The cumulative return aggregates future rewards as $R_t = \sum_{k=0}^{H} \gamma^k r_{t+k}$. where H represents the horizon and $\gamma \in (0, 1]$ is the discount factor. The discount factor γ reflects that LLM-based planners often fail to incorporate distant feedback due to context window limits and attention decay, effectively reducing the influence of temporally distant rewards. For infinite horizons, the discount factor ensures convergence of the cumulative return. Since rewards are bounded with $|r_t| \leq 1$, for any $\gamma \in (0, 1)$ we have:

$$|R_t| \leq \sum_{k=0}^{H} \gamma^k |r_{t+k}| \leq \sum_{k=0}^{H} \gamma^k \leq \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma} < \infty \tag{4}$$

This geometric series converges for $\gamma < 1$. For finite horizons $H = T - t$, the bound becomes $|R_t| \leq \frac{1-\gamma^{T-t+1}}{1-\gamma}$.

1) Critic: The critic $LLM_{crit.}$ with parameters ϕ estimates the value of the current dialog state, analogous to the A2C framework. Conditioned on the state s_t and short-term memory ST_m, which stores recent states, actions, and rewards, it leverages $LLM_{act.}$ as the policy to generate the next action. The cumulative return R_t summarizes expected future rewards, indicating the quality of the recent interaction history. Accordingly, the critic estimates the expected return of the current state as:

$$V_\phi(s_t) = \mathbb{E}_{a_t^i \sim LLM_{act.}}[R_t \mid s_t, ST_m] \tag{5}$$

where $V_\phi(s_t)$ serves as a baseline reference for evaluating future decisions. It does not change actions but serves as a baseline for evaluating corrections proposed by the critic $LLM_{crit.}$, enabling advantage estimation to assess whether a modification improves performance.

2) Advantage Verifier: Complementing the critic, the verifier $LLM_{adv.}$ with parameters ψ estimates the advantage and advantage and decides whether to accept a revision proposed by the critic. Given s_t, the original action a_t^i from $LLM_{act.}$, and adjustment Δa_t^i from $LLM_{crit.}$, it evaluates the revision via an advantage score:

$$\hat{A}_t^i = Q_\psi\big(s_t,\, a_t^i + \Delta a_t^i\big) - V_\phi(s_t) \tag{6}$$

where a higher $\hat{A}_t^i$ indicates that the adjusted action is expected to yield a better outcome than the original policy output. where $Q_\psi\big(s_t,\, a_t^i + \Delta a_t^i\big)$ approximates the immediate task success and feasibility of executing modified action in the current environment, and $V_\phi(s_t)$ is the critic's baseline estimate. The verifier checks whether the critic's revision improves task performance. If $\hat{A}_t^i > 0$, the correction is valid and a structured success summary (*e.g.*, "AGENT is [CHANGE] and is closer to [NAME OF OBJECT]") is stored in long-term memory LT_m for conditioning future action proposals by $LLM_{act.}$. If $\hat{A}_t^i \leq 0$, the revision is discarded due to constraint violations (*e.g.*, inverse kinematics errors), and $LLM_{act.}$ replans while retaining the original action. Accepted revisions are appended to ST_m and communicated to $LLM_{act.}$, ensuring that only corrections with positive advantage values ($\hat{A}_t^i > 0$) influence subsequent action generation and coordination. The advantage verifier $\text{LLM}_{adv.}$ evaluates whether a proposed correction Δa_t^i improves upon the original action a_t^i given state s_t and feedback r_t. The binary criterion ($\hat{A}_t^i > 0$) reduces this to a grounded feasibility check [28], while failed corrections propagate back through the critic into memory, enabling implicit self-correction across rounds.

3) Memory Consolidation: After each interaction cycle, the dialog history is stored in the long-term memory buffer LT_m to maintain prompt lengths while preserving essential context. At a specific time step t, the short-term buffer ST_t contains the current interaction round, including the latest states, actions, and rewards, while ST_{t-1} stores the previous interaction round. The short-term memory module ST_m manages these buffers by merging consecutive windows to maintain continuity across dialog turns. The resulting long-term memory update is defined as:

$$LT_t = \texttt{Merge}\big(ST_{t-1}, ST_t\big), \tag{7}$$

All interaction records are retained within LT_m and made available for future reasoning. The next actor prompt at time t is constructed as:

$$P_t = \texttt{Construct}\big(s_t,\, g,\, LT_m\big), \tag{8}$$

providing $LLM_{act.}$ with current state s_t, goal g, and accumulated dialog necessary to guide subsequent decisions. This cycle improves action quality and controls dialog length. Modular actor, critic, and verifier roles stabilize coordination, while structured memory grounds the decision process.

D. Prompt Design

1) Execution Prompt Design: We adopt the structured prompt format from RoCo [7], providing environment description, team objective, and reachable objects per robot. Robots respond sequentially, concatenated into execution triples (*e.g.*, `NAME Alice ACTION PICK OBJECT blue_cube NAME Bob ACTION MOVE OBJECT dustpan` **`EXECUTE`**). A validator checks action feasibility via inverse kinematics before execution. Our key contribution is restructuring prompt composition through memory integration and A2C feedback. The Actor prompt in Equation 2 now incorporates: long-term memory LT_m storing dialog history, execution outcomes, and verified corrections from the two most recent planning cycles; short-term memory ST_m injecting immediate inverse kinematics failures and collision detection reports; and retrospection analysis from Critic $LLM_{crit.}$, which diagnoses failure modes specifying which execution components failed (*e.g.*, inverse kinematics succeeded but collision detected, or trajectory collision-free but inverse kinematics infeasible). This A2C-based architecture with short-term and long-term memory integration enables Actor $LLM_{act.}$ to generate corrective action parameters informed by both historical execution patterns and Critic-verified failure diagnoses, with Advantage Verifier $LLM_{adv.}$ ensuring corrections satisfy $\hat{A}_t^i > 0$ before execution. Unlike RoCo [7], our approach incorporates memory integration and A2C-verified corrections for systematic improvement across planning cycles.

2) A2C-collab Prompt Design: To support effective instruction generation for multi-robot collaboration, we adopt A2C-Collab to guide LLMs in diagnosing failures and proposing improvements. Unlike RoCo [7], which employs feedback for error detection and replanning without validating correction quality, and [5], which introduces an actor-critic formulation that heuristically accepts critic feedback without formal validation, our approach establishes a rigorous verification stage through advantage estimation, providing a mathematically grounded criterion ($\hat{A}_t^i > 0$) to determine whether proposed corrections will improve performance, rather than relying on unchecked suggestions. Compared with [5], our framework formally defines the advantage verification process: at each cycle t, the actor $LLM_{act.}$ generates actions $a_t^i \sim \pi_\theta(a \mid s_t, p_t)$, the critic $LLM_{crit.}$ estimates the baseline $V_\phi(s_t) = \mathbb{E}_{a_t^i \sim LLM_{act.}}[R_t \mid s_t, ST_m]$ and proposes corrections Δa_t^i. Critically, our framework introduces the verifier $LLM_{adv.}$, which is entirely absent in [5], to evaluate corrections via $\hat{A}_i^t = Q_\psi(s_t, a_t^i + \Delta a_t^i) - V_\phi(s_t)$, accepting only those with $\hat{A}_t^i > 0$. Without this verification component, [5] lacks a mechanism to validate whether critic corrections actually improve performance. After dialog concludes with **EXECUTE**, the environment provides feedback on action plan feasibility. If actions fail due to unreachable objects or overlapping waypoints, the environment returns failure-specific messages, initiating a refinement process through three coordinated prompts:

Action Prompt. The $LLM_{act.}$ generates candidate actions based on the current state, task goal, environment description, and two recent dialog rounds, as shown below:

```
==== SYSTEM PROMPT ====
You are Alice, collaborating with Bob.
Task: Sweep cubes into dustpan.
[Capability] Actions: MOVE, PICK,
PLACE, SWEEP, WAIT
Equipment: Robotiq gripper; Workspace:
Left side of table
[Protocol] End with: PROCEED or EXECUTE
+ action plan
==== HISTORY ====
t-2: Alice moved to Blue_Cube; Bob
moved to dustpan
t-1: Alice picked Blue_Cube; Bob waited
==== CURRENT ROUND ====
Your gripper: holding Blue_Cube; Bob:
near dustpan, empty
Orange_Cube on table (your reach)
[Bob]: Should I catch it or you place?
Can you reach Orange?
You are Alice. Your response:
```

Critic Prompt. Upon receiving environment feedback, $LLM_{crit.}$ diagnoses failures by reviewing the previous dialog, as shown below:

```
[Previous action]
Alice: MOVE Blue_Cube Bob: MOVE Dustpan
[Feedback]
Execution failed: Alice is in the
waypoint of Bob, causing a collision.
Analyze why the previous plan failed
and propose a possible correction.
```

Advantage Prompt. The $LLM_{adv.}$ then evaluates the expected outcome of the proposed correction:

```
[Original plan]
Alice: MOVE Blue_Cube Bob: MOVE Dustpan
[Feedback]
Execution failed: Collision detected.
```

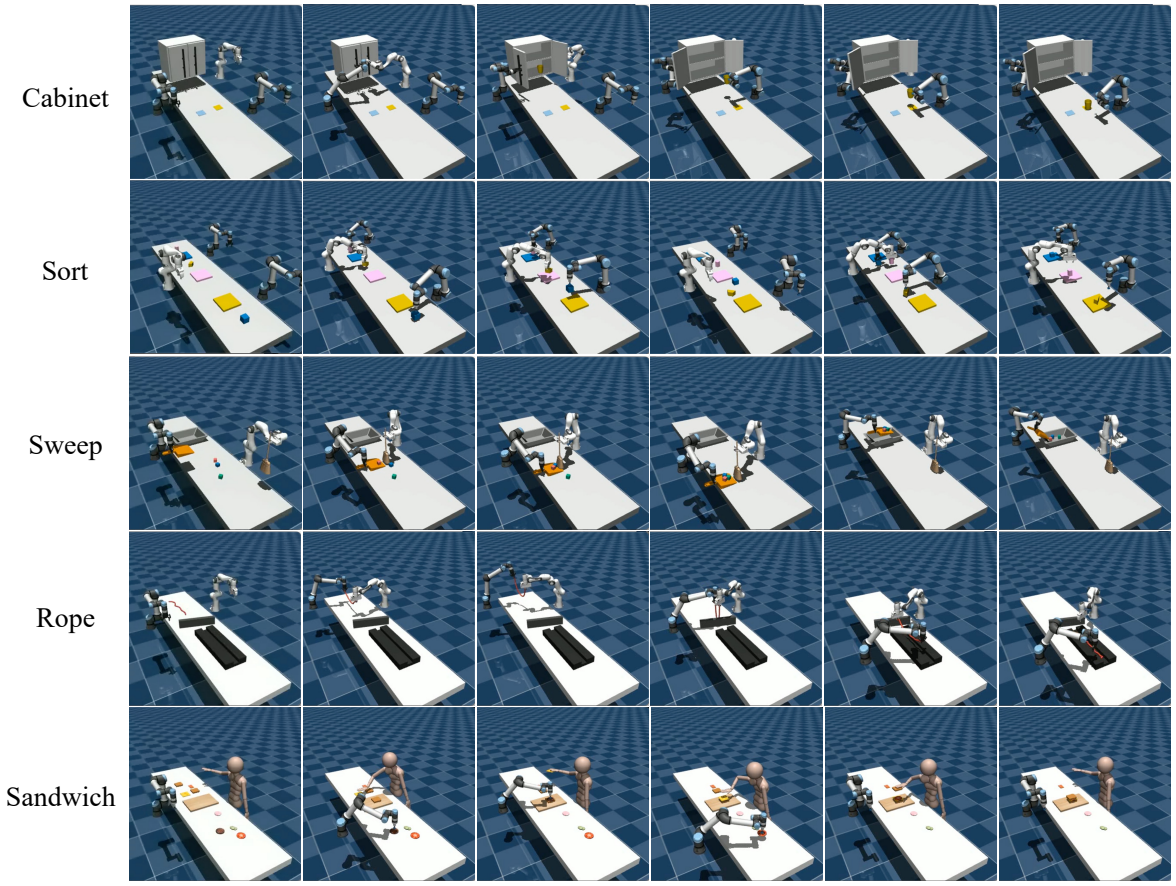

Fig. 2. Simulation results. We present the execution of all five tasks completed by the multi-robot system. Each row highlights the key steps performed by the designated robots to accomplish the task.

```
[Proposed correction from Critic]
Alice should wait until Bob reaches the
dustpan position first.
Evaluate the expected outcome if we
apply this correction.
Respond in the format: AGENT OBJECT.
```

IV. Experiments

To validate A2C-Collab, we conduct comprehensive experiments in a multi-robot simulation environment [7] with the following task objectives:

- ***Cabinet:*** Three robots collaborate to open a cabinet and place a cup on a mat, with two manipulating the door and one handling the cup.
- ***Sweep:*** Two robots cooperate to sweep three cubes into a dustpan and transfer them to a gray tray.
- ***Sandwich:*** Two robots collaborate to assemble a sandwich by sequentially picking and stacking ingredients in the correct order.
- ***Sort:*** Three robots collaborate to place three colored cubes onto their respective color-matched pads.
- ***Rope:*** Two robots lift a rope over a raised barrier and place it into a tray on the opposite side.

A2C-Collab employs Llama3.1-70B [29] as actor $LLM_{act.}$ and Llama3.2-3B [29] as critic $LLM_{crit.}$ and verifier $LLM_{adv.}$. Baselines include RoCo [7] with Llama3.1-70B replacing GPT-4 [30], and [5] using Llama3.1-70B/8B for actor/critic. Each task is executed 15 times, with success rates and episode lengths reported in Table I. All experiments use three NVIDIA A100 (80GB) GPUs; computational costs are reported in Table II.

As shown in Table I, A2C-Collab improves overall task success rates without increasing steps or replans. Here, 'replans' denote the number of dialog rounds in which the robots fail to reach agreement and do not produce a finalized `EXECUTE` command. Averaged over all tasks, A2C-Collab achieves a success rate of 0.39, compared with 0.31 for the actor-critic baseline and 0.21 for RoCo [7]. The largest gain appears in *Arrange Cabinet*, where success increases from 0.40 to 0.53 and mean steps decrease from 8.1 to 7.4, while replans remain near 3.0. *Move Rope* shows a similar improvement, with success rising from 0.33 to 0.40. In *Sweep Floor*, success improves from 0.20 to 0.27 without increasing replans. *Make Sandwich* and *Sort Cubes* also show consistent gains. Overall, incorporating advantage into the updates improves coordination while keeping steps and replans similar.

A. Real-world Experiments

We validate A2C-Collab in real-world experiments using two robotic arms: UFactory 850 (6 DoF) and UFactory 7 (7 DoF) [31], as shown in Figure 3. Tasks require collaboration through workspace constraints and action restrictions. Scene

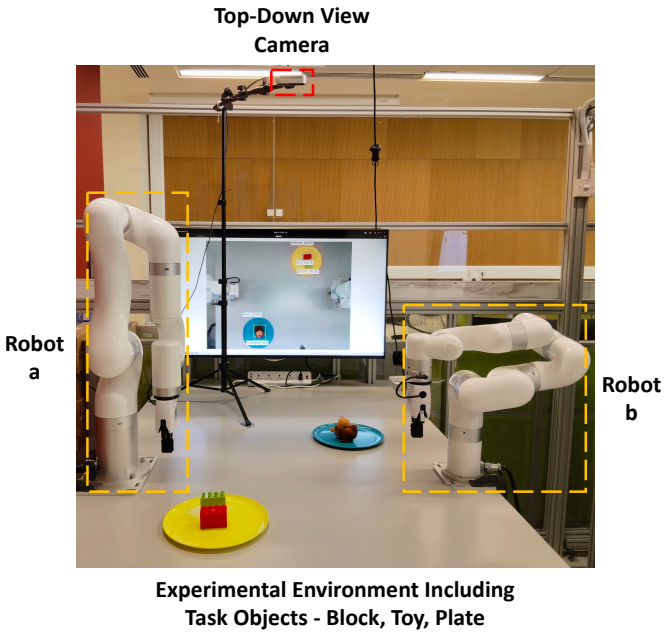

Fig. 3. Real-world experiment setting. The setup consists of two collaborative robot arms ('Robot a' and 'Robot b') with a top-down view camera for scene observation and task objects in a shared workspace.

descriptions are generated using OWL-ViT [32] from top-down RGB-D images captured by a RealSense camera [33]. The tasks are described as follows:

- ***Sort Block and Toy:*** In this task, 'Robot a' sorts toys into Plate 1 (blue), while 'Robot b' sorts blocks into Plate 2 (yellow), each assigned to a distinct workspace.
- ***Move Block:*** This task requires 'Robot a' to pick blocks from a cardboard box (brown) and place them on the paper area (white), after which 'Robot b' transfers the blocks to the plate (yellow).

As shown in Table III, each task was conducted 10 times,

TABLE I
RESULTS ON ROCOBENCH [7]: SUCCESS RATE (↑), STEPS (↓), AND RE-PLANS (↓) OVER 15 RUNS PER TASK.

Task	RoCo [7]	Actor-Critic [5]	A2C-collab
Arrange Cabinet	0.27±0.12	0.40±0.13	**0.53**±0.13
(Steps, Replans)	8.9, 3.1	8.1, 3.1	**7.4, 3.0**
Sweep Floor	0.07±0.07	0.20±0.10	**0.27**±0.11
(Steps, Replans)	9.9, 1.1	9.7, 1.1	**9.2**, 1.2
Make Sandwich	0.13±0.09	0.20±0.10	**0.33**±0.12
(Steps, Replans)	**8.3**, 2.3	9.7, **2.2**	9.2, 2.6
Sort Cubes	0.33±0.12	**0.40**±0.13	**0.40**±0.13
(Steps, Replans)	8.9, 3.1	8.1, **3.0**	**8.0**, 3.5
Move Rope	0.27±0.12	0.33±0.12	**0.40**±0.13
(Steps, Replans)	5.6, 4.2	**4.6, 3.8**	6.9, 4.0

TABLE II
COMPUTATIONAL COST: AVERAGE RUNTIME AND GPU MEMORY.

	Sort	Cabinet	Rope	Sweep	Sandwich
Time (min)	11.1±1.7	5.4±2.4	17.9±5.3	5.5±1.3	14.7±0.4
GPU (GB)	147.9	150.9	144.7	158.4	149.8

TABLE III
REAL-WORLD EVALUATION ON SORT BLOCK & TOY AND MOVE BLOCK TASKS (EXECUTIONS / SUCCESSES OUT OF 10 TRIALS ↑).

Task	Exec. (10)	Success (10)	Rate (%)
Sort Block & Toy	10/10	9/10	90.0
Move Block	9/10	9/10	100.0
Average	9.5/10	9/10	95.0

evaluating executable rate (whether the model generates a valid plan) and success rate (whether actions complete successfully). Our framework achieved 0.95 executable rate and 0.9 success rate on average. Qualitative results in Figure 4 demonstrate successful action plan generation for both robotic arms.

TABLE IV
ABLATION: SINGLE-LENGTH MEMORY VS. OURS

Task	Single Length Memory Success, Steps, Replan	Ours (A2C-collab) Success, Steps, Replan
Arrange Cabinet	0.33±0.12, 7.9, 3.2	**0.53**±0.13, **7.4**, **3.0**
Sweep Floor	**0.27**±0.11, 9.3, **1.2**	**0.27**±0.11, **9.2**, **1.2**
Make Sandwich	0.07±0.07, 10.0, 5.0	**0.33**±0.12, **9.2**, **2.6**
Sort Cubes	0.07±0.07, 9.8, 3.8	**0.40**±0.13, **8.0**, **3.3**
Move Rope	**0.40**±0.13, 7.3, 3.9	**0.40**±0.13, **4.6**, **3.8**

V. ABLATION STUDY

We perform ablation studies analyzing design choices: a comparison of memory length between single round memory and our approach, followed by five model variations to assess architectural factors. All tasks are tested over 15 rollouts.

A. Memory Length

We compare our two-round memory design against a single-round memory setting, with the results reported in Table IV. This analysis illustrates the contribution of the proposed memory design to task performance.

Single-length memory lead to noticeable drops in success rates, particularly in multi-step tasks such as *Make Sandwich* (0.07 vs. 0.33) and *Sort Cubes* (0.07 vs. 0.40), where longer memory provides substantial advantages. Simpler tasks like *Sweep Floor* show minimal difference. Longer memory also improves efficiency: in *Move Rope*, steps decrease from 7.3 to 4.6 while success rates remain unchanged. However, extended context increases VRAM usage, requiring balance between historical information and computational constraints.

B. Model Alteration

To investigate the impact of model size, we conduct an ablation study on A2C-collab by replacing the critic ($LLM_{crit.}$) and advantage verifier ($LLM_{adv.}$) with smaller models while keeping the actor ($LLM_{act.}$) fixed, except in one setting where all three agents are replaced. As summarized in Table V, we evaluate five configurations. First, we replace both $LLM_{crit.}$ and $LLM_{adv.}$ with DeepSeekCoder-1.3B [34] (Coder). Second, we replace both with Qwen2.5-1.5B-Instruct [35] (Qwen). Third, we replace them with

TABLE V
ABLATION STUDY: DIFFERENT MODEL COMBINATIONS

Task	Coder+Coder Succ., Step, Repl.	Qwen+Qwen Succ., Step, Repl.	Llama+Gemma Succ., Step, Repl.	Coder+Gemma Succ., Step, Repl.	Llama-3B Succ., Step, Repl.	Ours (A2C-collab) Succ., Step, Repl.
Cabinet	0.20±0.10, 9.6, 3.0	0.33±0.12, 8.3, 3.3	0.33±0.12, 7.6, 3.2	0.20±0.10, 9.6, **2.9**	0.07±0.07, 9.5, 4.7	**0.53**±0.13, **7.4**, 3.0
Sweep	0.20±0.10, **9.3**, 1.6	0.20±0.10, 9.6, **1.0**	0.20±0.10, 9.5, 1.2	0.13±0.09, 9.5, 1.1	0.13±0.09, 10.0, 2.8	**0.27**±0.11, 9.2, 1.2
Sandwich	0.07±0.07, 9.6, 5.0	0.07±0.07, 9.4, 5.0	0.13±0.09, 9.7, 4.9	0.07±0.07, **7.6**, 4.9	0.07±0.07, 10.0, 5.0	**0.33**±0.12, 9.2, **2.6**
Sort	**0.40**±0.13, 8.0, **2.9**	0.20±0.10, 9.5, 3.3	0.13±0.09, 9.7, 3.6	0.27±0.11, 9.3, 3.0	0.07±0.07, 9.9, 4.6	**0.40**±0.13, **8.0**, 3.3
Rope	0.33±0.12, 7.1, **3.4**	0.27±0.11, 9.0, 3.9	0.26±0.11, 9.1, 4.2	0.27±0.11, 6.2, 4.0	0.13±0.09, 8.6, 5.0	**0.40**±0.13, **4.6**, 3.8

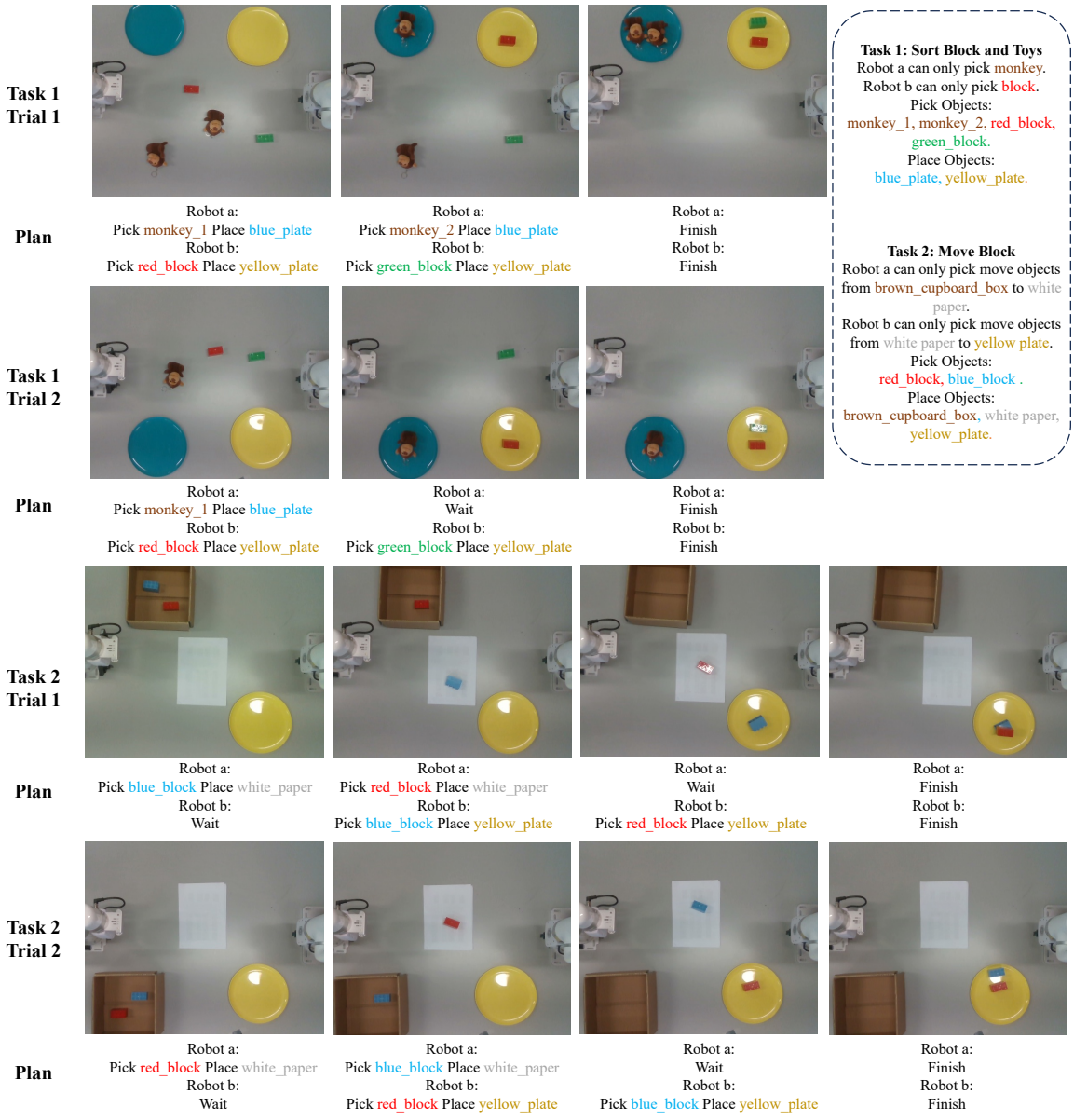

Fig. 4. Real-world qualitative results. Generated plans and corresponding robot actions for each step on Task 1 and Task 2 across two trials, demonstrating the consistency and robustness of the proposed method in real-world environments.

Llama-3.2-3B-Instruct [29] (Llama-3B) and Gemma2-2B-Instruct [36] (Gemma), respectively. Fourth, we replace them with DeepSeekCoder-1.3B [34] (Coder) and Gemma2-2B-Instruct [36] (Gemma), respectively. Finally, we replace all three agents with Llama-3.2-3B-Instruct [29].

Smaller backbones struggled to achieve consistent success rates, with Qwen+Qwen and Coder+Coder variants stabilizing at 0.20–0.33 on *Cabinet* and only 0.07 on *Sandwich*. Additionally, the Llama-7B [29] with Coder [34] configuration performed worst overall. A2C-Collab is more stable on complex tasks (*Sandwich*, *Rope*), showing the need for larger models in reasoning and verification.

VI. Conclusion

We introduced A2C-Collab, an advantage actor-critic framework for multi-robot collaboration, evaluated in simulated and real-world settings. Results show improvements in success rate and coordination, demonstrating that advantage evaluation enhances stability and cooperative decisions. However, LLM reasoning remains vulnerable to hallucinations and prompt sensitivity. Future work will focus on fine-tuning and dialog consistency to improve reliability.

Acknowledgment

The authors appreciate the support by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010.

References

[1] Y. Yang, Y. Lyu, Y. Zhang, S. Yi, and W. Luo, "Decentralized multi-robot line-of-sight connectivity maintenance under uncertainty," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[2] L. Heuer, L. Palmieri, A. Mannucci, S. Koenig, and M. Magnusson, "Benchmarking multi-robot coordination in realistic, unstructured human-shared environments," in *IEEE International Conference on Robotics and Automation*. IEEE, 2024, pp. 14 541–14 547.

[3] C. Azevedo, P. U. Lima, B. Lacerda, and N. Hawes, "Formal and scalable multi-robot coordination methods for long horizon tasks with time uncertainty," *Robotics and Autonomous Systems*, vol. 193, p. 105103, 2025.

[4] G. Sriramanan, S. Bharti, V. S. Sadasivan, S. Saha, P. Kattakinda, and S. Feizi, "Llm-check: Investigating detection of hallucinations in large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 34 188–34 216, 2024.

[5] J. Liang, H. Huang, Y. Hao, G. C. R. Bethala, C. Wen, and Y. Fang, "Integrating retrospective framework in multi-robot collaboration," in *International Conference on Automation, Robotics, and Applications*. IEEE, 2025, pp. 195–199.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1928–1937.

[7] Z. Mandi, S. Jain, and S. Song, "Roco: Dialectic multi-robot collaboration with large language models," in *IEEE International Conference on Robotics and Automation*. IEEE, 2024, pp. 286–299.

[8] X. Lin, Y. Huang, F. Chen, and B. Englot, "Decentralized multi-robot navigation for autonomous surface vehicles with distributional reinforcement learning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2024, pp. 8327–8333.

[9] A. Tajbakhsh, L. T. Biegler, and A. M. Johnson, "Conflict-based model predictive control for scalable multi-robot motion planning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2024, pp. 14 562–14 568.

[10] A. Moldagalieva, J. Ortiz-Haro, M. Toussaint, and W. Hönig, "Db-cbs: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2024, pp. 14 569–14 575.

[11] M. Fernando, R. Senanayake, H. Choi, and M. Swany, "Graph attention multi-agent fleet autonomy for advanced air mobility," *arXiv preprint arXiv:2302.07337*, 2023.

[12] W. Liu, W. Jing, K. Guo, G. Xu, Y. Liu *et al.*, "Traco: Learning virtual traffic coordinator for cooperation with multi-agent reinforcement learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 2465–2477.

[13] R. T. Fawcett, L. Amanzadeh, J. Kim, A. D. Ames, and K. A. Hamed, "Distributed data-driven predictive control for multi-agent collaborative legged locomotion," in *IEEE International Conference on Robotics and Automation*. IEEE, 2023, pp. 9924–9930.

[14] M. Tzes, N. Bousias, E. Chatzipantazis, and G. J. Pappas, "Graph neural networks for multi-robot active information acquisition," in *IEEE International Conference on Robotics and Automation*. IEEE, 2023, pp. 3497–3503.

[15] Z. Xu, X. Lin, and V. Tzoumas, "Bandit submodular maximization for multi-robot coordination in unpredictable and partially observable environments," in *Robotics: Science and Systems*, 2023.

[16] S. Singh, A. Srikanthan, V. Mallampati, and H. Ravichandar, "Concurrent constrained optimization of unknown rewards for multi-robot task allocation," *arXiv preprint arXiv:2305.15288*, 2023.

[17] L. C. Bezerra, A. M. Dos Santos, and S. Park, "Learning policies for dynamic coalition formation in multi-robot task allocation," *IEEE Robotics and Automation Letters*, 2025.

[18] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2019.

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

[20] 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 *IEEE International Conference on Robotics and Automation*. IEEE, 2023, pp. 9493–9500.

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

[22] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[23] F. Joublin, A. Ceravola, P. Smirnov, F. Ocker, J. Deigmoeller, A. Belardinelli, C. Wang, S. Hasler, D. Tanneberg, and M. Gienger, "Copal: corrective planning of robot actions with large language models," in *IEEE International Conference on Robotics and Automation*. IEEE, 2024, pp. 8664–8670.

[24] 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.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[27] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: a physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[28] S. Hao, Y. Gu, H. Ma, J. Hong, Z. Wang, D. Wang, and Z. Hu, "Reasoning with language model is planning with world model," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 8154–8173.

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

[30] 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.

[31] "Cost-effective cobot robots," UFactory, 2023, available online: https://www.ufactory.cc.

[32] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen *et al.*, "Simple open-vocabulary object detection with vision transformers. arxiv 2022," *arXiv preprint arXiv:2205.06230*, vol. 2, 2022.

[33] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1–10.

[34] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li *et al.*, "Deepseek-coder: When the large language model meets programming–the rise of code intelligence," *arXiv preprint arXiv:2401.14196*, 2024.

[35] Q. Team, "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115*, 2025.

[36] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024.