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

ActionReasoning: Robot Action Reasoning in 3D Space with LLM for Robotic Brick Stacking

Guangming Wang*, Qizhen Ying*, Yixiong Jing†, Olaf Wysocki, and Brian Sheil

***Abstract*— Classical robotic systems typically rely on custom planners designed for constrained environments. While effective in restricted settings, these systems lack generalization capabilities, limiting the scalability of embodied AI and general-purpose robots. Recent data-driven Vision-Language-Action (VLA) approaches aim to learn policies from large-scale simulation and real-world data. However, the continuous action space of the physical world significantly exceeds the representational capacity of linguistic tokens, making it unclear if scaling data alone can yield general robotic intelligence. To address this gap, we propose ActionReasoning, an LLM-driven framework that performs explicit action reasoning to produce physics-consistent, prior-guided decisions for robotic manipulation. ActionReasoning leverages the physical priors and real-world knowledge already encoded in Large Language Models (LLMs) and structures them within a multi-agent architecture. We instantiate this framework on a tractable case study of brick stacking, where the environment states are assumed to be already accurately measured. The environmental states are then serialized and passed to a multi-agent LLM framework that generates physics-aware action plans. The experiments demonstrate that the proposed multi-agent LLM framework enables stable brick placement while shifting effort from low-level domain-specific coding to high-level tool invocation and prompting, highlighting its potential for broader generalization. This work introduces a promising approach to bridging perception and execution in robotic manipulation by integrating physical reasoning with LLMs.**

I. INTRODUCTION

Recent progress in humanoids and general manipulation policies has attracted significant attention in embodied intelligence. VLA models, such as RT-2 [1] and OpenVLA [2], large open datasets like Open-X-Embodiment [3], and open policies such as Octo [4] demonstrate that scaling data and model capacity can improve reusability across tasks and hardware platforms. However, while LLMs in the text domain demonstrate emergent generalization through scaling, such "scale→emergence" capability has not yet been achieved with comparable robustness for universal robot control. Performance continues to degrade across diverse tasks, embodiments, and long-horizon reasoning, which indicates that merely scaling data and parameters alone is insufficient for robust generalization. Therefore, the proposition that "sheer scaling alone can yield general-purpose manipulation" needs to be treated with increasing caution in the robotics community, as there is a much larger solution space of robot actions compared to the language space.

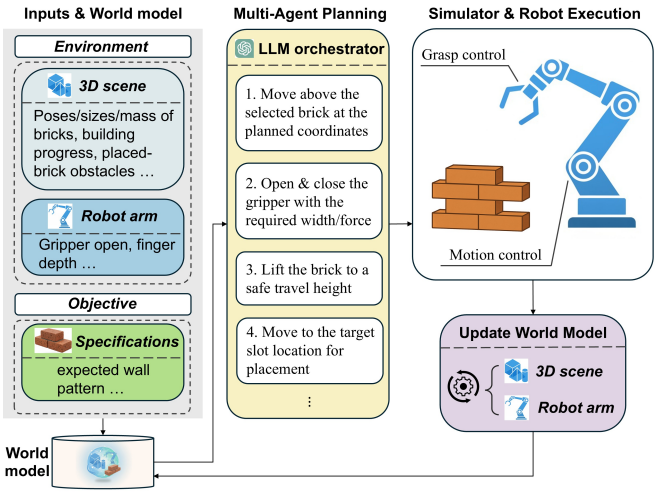

Fig. 1: Three stages of the ActionReasoning pipeline. (1) Inputs & World Model: The world model of bricklaying is provided as input. (2) Multi-Agent Planning: Leveraging the world model input, an LLM orchestrator decomposes the task into specialized agents that generate actions and waypoints to plan the motion of a selected brick toward its target location. (3) Simulator & Robot Execution: The robot (a Kuka simulator in this case) executes the planned actions from multiple agents to control grasp and motion. Observations from changes in the 3D scene and robot arm state are used to update the world model, enabling continual re-planning as the task progresses.

The lack of generalizability in robotics has motivated the increasing attention and research interest in world models. World models have been defined and studied in multiple ways. Some research focuses on predicting environmental dynamics and deducing future states in a latent space to support planning and control [5], [6]. Other recent works have introduced 'generative foundation world models' [7], [8] that produce interactive environments from condition information, like one image or one sentence of the current environment. The term spans both an abstraction of universal physical laws and commonsense knowledge, as well as computable representations of scene geometry and dynamics in the current environment. Accordingly, this paper adopts

This work was supported in part by Computer Vision for Digital Twins (CV4DT), Cambridge Centre for Smart Infrastructure and Construction (CSIC) and Laing O'Rourke Centre for Construction Engineering and Technology, Cambridge. (Corresponding Author: Yixiong Jing)

* Equal contributions † Co-corresponding authors

G. Wang, Y. Jing, Olaf Wysocki, and B. Sheil are with the CV4DT, CSIC, Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, U.K. (e-mail: gw462@cam.ac.uk, yj401@cam.ac.uk, okw24@cam.ac.uk, bbs24@cam.ac.uk)

Q. Ying is with the Department of Engineering, University of Oxford, Oxford, U.K. (e-mail: qizhen.ying@exeter.ox.ac.uk).

Code will be available at https://github.com/StephenYing/Action_Reasoning

an operational definition for robots: World Model = (A) universal physical knowledge and commonsense priors + (B) precise representations of the environment in which the robot operates. Here, (A) corresponds to a "general knowledge base," while (B) aligns with Simultaneous Localization and Mapping (SLAM) [9] and semantic geometry [10], where (B) enables the environment state to be queryable and constrained.

This paper focuses on how a world model can be used for robotics tasks, rather than how to learn one. As shown in Fig. 1, we assume that the robot can obtain accurate 3D environmental states (e.g., 3D scene and robot arm states) through existing perception pipelines, and then combine these with the physical commonsense and task specifications (e.g., objective) already internalized in LLMs. This integration enables a form of 3D action reasoning: the LLM performs multi-step physical reasoning in the 3D environment with world model input, where the reasoning from the LLM orchestrator guides the robot arm in the simulator to finish tasks. To make the problem concrete and testable, we conduct systematic experiments and ablations on a representative task: brick stacking. This task involves typical physical constraints such as object contact, stability, and tolerance, while also requiring multi-step sequencing and precise pose coordination, making it a suitable testbed for 3D action reasoning.

The recent LLM-based robot system, ReKep [11], leveraged a generalizable perception models, Vision Language Models (VLMs), to identify actionable keypoints in the environment. However, 2D action keypoints lack understanding of complex 3D scenes, making them vulnerable to occlusion, foreshortening, and varied camera placements. Moreover, when LLMs are given images as inputs, these keypoints occupy a certain image region and are sometimes ambiguous or poorly localized. In addition, it still relies on dedicated off-the-shelf solvers. Our framework differs in that it (i) ingests explicit 3D scene states as LLM input, reducing reliance on 2D keypoint without 3D sensing, (ii) outputs phase-conditioned action proposals and performs multi-turn LLM calls when needed to refine plans, mimicking humans' "think-while-doing".

Overall, this work achieves the following contributions:

- We introduce ActionReasoning, i.e., a 3D action-reasoning framework that enables robot decision-making for brick stacking with high-level tool invocation and prompting, demonstrating generalization across configurations without per-scene code.
- We design a multi-agent LLM orchestrator, where the specialized agents at different stages fuse human task specifications with world model input to reason in SE(3).
- We provide simulation studies and ablations showing that 3D LLM reasoning improves robustness compared with the traditional control method with a similar simple programming, establishing a foundation for future LLM-driven physical reasoning in robotics.

II. RELATED WORK

A. Robotic Manipulation Learning

Traditional manipulation robots rely on a structured pipeline in which the scene is perceived, a task-level representation is constructed, and detailed motions are programmed for execution within a given environment. Task-and-motion planning (TAMP) integrates symbolic task decomposition with continuous motion feasibility, but still assumes substantial domain modelling and engineering for each setting [12]. Even seemingly simple stacking tasks require extensive development on scripted sequencing, grasp poses, and constraint handling.

Subsequently, Reinforcement Learning (RL) enabled training in environments without the need for task-specific programming for grasping or manipulation. By employing an action policy network and designing appropriate rewards, a robotic arm can explore the action space within simulations and achieve target tasks rewarded by the reward function. Since object positions (e.g., brick placements) can be freely configured in simulation, the success rate in simulated environments is often high. The subsequent research began exploring ways to accelerate this process, for example, by learning a corrective policy on top of a base controller [13] or using base controllers as priors to guide exploration [14]. Such controllers bias the search toward regions of the solution space where conventional algorithms are more likely to succeed, thereby speeding up RL convergence. However, this still requires manually designing and coding the low-level controllers, with the ultimate aim of enabling RL to converge faster and eventually outperform the controllers themselves. Despite these advances, the sim-to-real gap remains a major challenge in this line of research [15].

Although the recent Sim2Real2Sim pipeline proposed in [15] has made such sim-to-real transfers feasible, such trained models remain limited to the sampled real-world data distribution. Training a general model which is deployable in the real world still demands extensive real data collection and 3D modelling efforts. Similarly, VLA models [1], [2], [4] push end-to-end learning from vision/language to actions, but also require large data. As a result, industry continues to rely heavily on large-scale data acquisition with demonstration datasets [16], [17] and teleoperation datasets [18], [19]. These remain the mainstream solutions for improving the general capabilities of robots. However, a key challenge is that the robots' action space is far larger than the vocabulary space of language. For LLMs, although the human vocabulary is vast, it is still finite. Once each word is vectorized, predicting the next word becomes a problem of discrete space prediction. In contrast, the action space of robots is enormously wide, raising the question of whether a scaling law exists and when it can arise for robots.

B. LLM/VLM Based Robotic Operation

Thanks to access to big data, LLMs have achieved a form of general natural language question answering and have acquired substantial human-like understanding. Some

might argue that LLMs have learned a mode of thinking similar to humans, while others insist that LLMs merely predict the distribution of the next token and essentially learn statistical patterns from data. The fact remains that LLMs can interpret and respond to general and intuitive questions. Moreover, they are increasingly capable of addressing domain-specific queries and drawing on broad knowledge of physics, mathematics, and other disciplines. The introduction of chain-of-thought reasoning has further enabled LLMs to follow human-preferred step-by-step reasoning processes in problem solving [20], [21]. These advances have greatly enhanced the capabilities of LLMs, making them valuable for supporting robotic tasks such as task understanding, planning, and reasoning.

VLA models integrate visual understanding, language reasoning, and action generation to achieve action-level reasoning [1], [2], [4]. However, end-to-end training requires massive datasets as we discussed in Section II-A. Moreover, the training also introduces uncertainty at every stage, which spans from visual perception to linguistic reasoning and to final action generation. To address this, we argue for decoupling the visual component: assuming the robot already has sufficiently accurate perceptual information via computer vision algorithms, and the LLMs are asked to focus only on action reasoning. This approach significantly reduces data requirements while also leveraging the wealth of existing research in computer vision. For example, the ReKep series [11], [22] use vision-language models to identify keypoints for robotic manipulation, which significantly reduces the reliance on hand-coded programs and does not require training. Such approaches also constrain the robot's exploration space. The distinction of our work from ReKep [11] is that while ReKep emphasizes 2D keypoint detection and corresponding actions, our method enables direct 3D spatial reasoning leveraging physical information. We therefore term it physical reasoning. This allows robots to exploit richer physical cues and directly operate in SE(3) to produce phase-conditioned action proposals in 3D, avoiding limitations such as occlusions and the lack of precision inherent in 2D keypoint inputs.

In addition, for complex multi-step tasks, there are emerging multi-agent frameworks, such as CAMEL [23] for role-playing agents and AutoGen [24] for multi-agent conversations, that enable the coordination of specialized agents to debate, verify, or decompose tasks. Embodied agents such as Voyager [25] demonstrate open-ended skill acquisition with iterative prompting and a growing skill library. These patterns inform our design of specialized robotics agents, task decomposition, physics checking, and pose planning, while collaborating over a shared and updated 3D environment state. Our method employs multiple LLM calls, where each call is assigned specific roles to decompose complex problems, enabling end-to-end reasoning based on LLMs [26].

III. METHODS

We aim to bridge perception and execution for robotic manipulation via physical reasoning with LLMs. The input is the robot's structured understanding of the current environment state, denoted $S_t \in \mathscr{S}$, including 3D scene geometry, brick poses, placed-brick obstacles, robot arm state, and task context, together with a goal specification $G \in \mathscr{G}$. The output is an executable action for the manipulator, which is named as a waypoint, representing the pose of end-effector $w_{t+1} \in \mathrm{SE}(3)$. Sometimes the output also includes the gripper opening and closing, but the following formula will omit this for the sake of simplicity. Low-level control, trajectory interpolation and high-rate control updates are handled by the robot controller embedded in the robot arm. Our objective is to use LLM with physics reasoning to close the gap between "understanding" and "doing" without a lot of low-level domain-specific coding.

A. LLM based Multi-agent Framework for Robot Action Reasoning

1) Markovian waypoint loop: Let $A_t \in \mathscr{A}$ denote an action applied at time t. The manipulator influences the environment through A_t, producing a transition $S_t \to S_{t+1}$. We run in waypoint mode: the method emits the next target pose $w_{t+1} \in \mathrm{SE}(3)$. After each waypoint is executed, perception is refreshed, and the reasoner is called again. This yields a Markovian closed loop whose sampling granularity K_t adapts to task precision.

$$w_{t+1} = \pi_{\mathrm{AR}}(S_t, G) \in \mathrm{SE}(3), \tag{1}$$

where LLM π_{AR} is used for 3D action reasoning to obtain the target pose w_{t+1} from current environment state S_t and goal G.

Then, the sequence of low-level control commands $\mathbf{u}_{t,1:K_t}$ are interpolated over K_t servo ticks to drive the robot from the current state x_t toward the target waypoint w_{t+1} as follows.

$$\mathbf{u}_{t,1:K_t} = \mathrm{Interp}\big(x_t \to w_{t+1}; K_t\big), \tag{2}$$

where $\mathrm{Interp}(\cdot)$ is the internal trajectory generator. This is completed in the robot arm with default parameters and is not an academic contribution of this paper. Influenced by $\mathbf{u}_{t,1:K_t}$, the environment transits from S_t to S_{t+1} by the world transition $\mathscr{T}$, which is simulated to reflect the physical laws in a real word.

$$S_{t+1} \sim \mathscr{T}\big(S_t, \mathbf{u}_{t,1:K_t}\big) \tag{3}$$

Then, the robot at pose x_{t+1} perceives the updated environment to refresh to the structured environment state $\hat{S}_{t+1}$.

$$\hat{S}_{t+1} = \Phi\big(\mathrm{Perceive}(x_{t+1})\big) \tag{4}$$

To focus on our contribution in physical reasoning with LLMs, we adopt the simplification $\hat{S}_{t+1} = S_{t+1}$.

Therefore, we obtain the Markov assumption that the next state depends only on the current state S_t and the chosen waypoint w_{t+1}, rather than the full history $S_{0:t}$, which can be represented as follows.

$$p(S_{t+1} \mid S_{0:t}, G) = p(S_{t+1} \mid S_t, w_{t+1}) \tag{5}$$

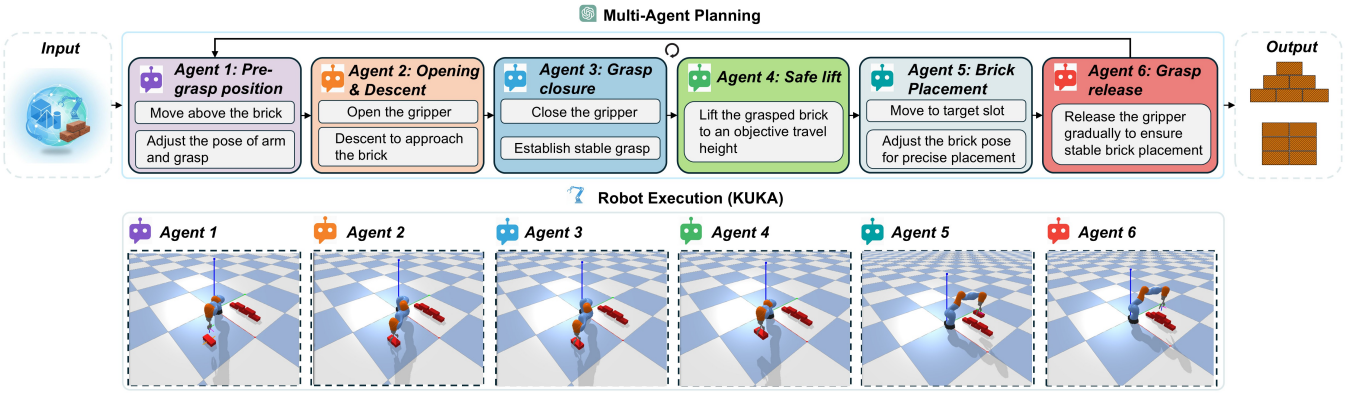

Fig. 2: Illustration of the six agents ($Ag_1 - Ag_6$) in the present ActionReasoning framework: (1) Pre-grasp positioning to guide the arm to approach the brick; (2) Opening and descent to position the gripper above the brick surface; (3) Grasp closure to secure the brick; (4) Safe lift to raise the brick from the ground; (5) Brick placement to stably move and accurately align the brick at the target location; and (6) Grasp release to land the brick. The corresponding execution of each agent is illustrated on the simulated KUKA robot arm at the bottom of this figure.

2) Feasible set and selection: We cast action selection as a physics-guided inverse problem: given $S_t \in \mathscr{S}$ and $G \in \mathscr{G}$, infer an action (or waypoint) that makes goal-directed progress while satisfying safety and feasibility. Let F be a prior physical deduction operator encoding collision avoidance, reachability, contact stability, and tolerances.

$$\begin{aligned}\mathscr{A}_t^{\text{feas}} &= F(S_t, G) \\ &= \Big\{ a \in \mathscr{A} \;\Big|\; \underbrace{\mathscr{C}_{\text{phys}}(S_t,a) \le 0}_{\text{physics/safety}},\ \underbrace{\mathscr{P}_{\text{reach}}(S_t,a) = 1}_{\text{reachability}}, \\ &\qquad \underbrace{\Delta(S_t,a;G) \le \varepsilon}_{\text{goal progress}} \Big\},\end{aligned} \tag{6}$$

$$a_t^\star = \arg\min_{a \in \mathscr{A}_t^{\text{feas}}} J(a; S_t, G), \tag{7}$$

where $\mathscr{C}_{\text{phys}}$ captures collision/contact limits, $\mathscr{P}_{\text{reach}}$ is a reachability predicate (including kinematic and joint limits), Δ measures residual error relative to G, ε is a tolerance, and J trades off path length, clearance, and alignment quality. In practice, the LLM proposes candidates consistent with priors, and a verifier prunes them using F.

3) Agent pipeline with gating: Long-horizon brick stacking is handled by a sequential gated multi-agent pipeline. Each agent Ag_i consumes the current state and upstream messages, outputs a message m_i (e.g., a pose proposal, constraints, or a waypoint), and returns an acceptance flag $\sigma_i \in \{0,1\}$. The next agent executes only if the previous one accepts.

As shown in Fig. 2, we instantiate six specialized agents Ag_1–Ag_6 that transform high-level intent into physically consistent motion targets. We can formulate them simply as:

$$(m_1, \sigma_1) = Ag_1(S_t, G), \tag{8}$$
$$(m_2, \sigma_2) = Ag_2(S_t, G, m_1), \tag{9}$$
$$(m_3, \sigma_3) = Ag_3(S_t, G, m_{1:2}), \tag{10}$$
$$(m_4, \sigma_4) = Ag_4(S_t, G, m_{1:3}), \tag{11}$$
$$(m_5, \sigma_5) = Ag_5(S_t, G, m_{1:4}), \tag{12}$$
$$(m_6, \sigma_6) = Ag_6(S_t, G, m_{1:5}), \tag{13}$$
$$A_t = \big(Ag_6 \circ Ag_5 \circ Ag_4 \circ Ag_3 \circ Ag_2 \circ Ag_1\big)(S_t, G) \quad \text{iff} \quad \sigma_i = 1 \ \forall i \in \{1,\dots,6\}. \tag{14}$$

The gating rule can be written as

$$Ag_{i+1} \text{ is executed iff } \ \sigma_i = \mathbf{1}\big[\text{Checks}_i\big(S_t, G, m_{1:i}\big)\big] = 1. \tag{15}$$

Specifically, for each agent, the design is as follows. Below, $\hat{n}$ denotes a surface normal, f_n is a measured normal force, μ is an estimated friction coefficient, and $\mathbf{e}_{xy}$ and e_θ denote the planar translation and yaw alignment errors, respectively.

a) Ag_1 Pre-grasp positioning: The agent moves the robot end-effector above the target brick and aligns the gripper approach vector with the brick's grasp axis. The agent proposes an approach pose $p_{\text{app}} \in \text{SE}(3)$ with clearance $c \ge c_{\min}$:

$$\sigma_1 = \mathbf{1}\big[\text{CollisionFree}\big(\text{Path}(x_t \to p_{\text{app}})\big) \wedge c \ge c_{\min}\big]. \tag{16}$$

b) Ag_2 Descent & opening: The agent opens the gripper to width $w \ge w_{\text{brick}} + \delta$ and descends along $-\hat{z}$ into a graspable pose $p_\downarrow$:

$$\sigma_2 = \mathbf{1}\big[w \ge w_{\text{brick}} + \delta \wedge \text{NoContactSidewalls}(p_\downarrow)\big]. \tag{17}$$

c) Ag_3 Grasp closure: The agent closes the gripper and verifies a stable contact using a normal-force threshold and pose tolerance:

$$\sigma_3 = \mathbf{1}\big[f_n \ge f_{\min} \wedge \text{PoseError}(p_\downarrow, p_{\text{grasp}}) \le \varepsilon_g\big]. \tag{18}$$

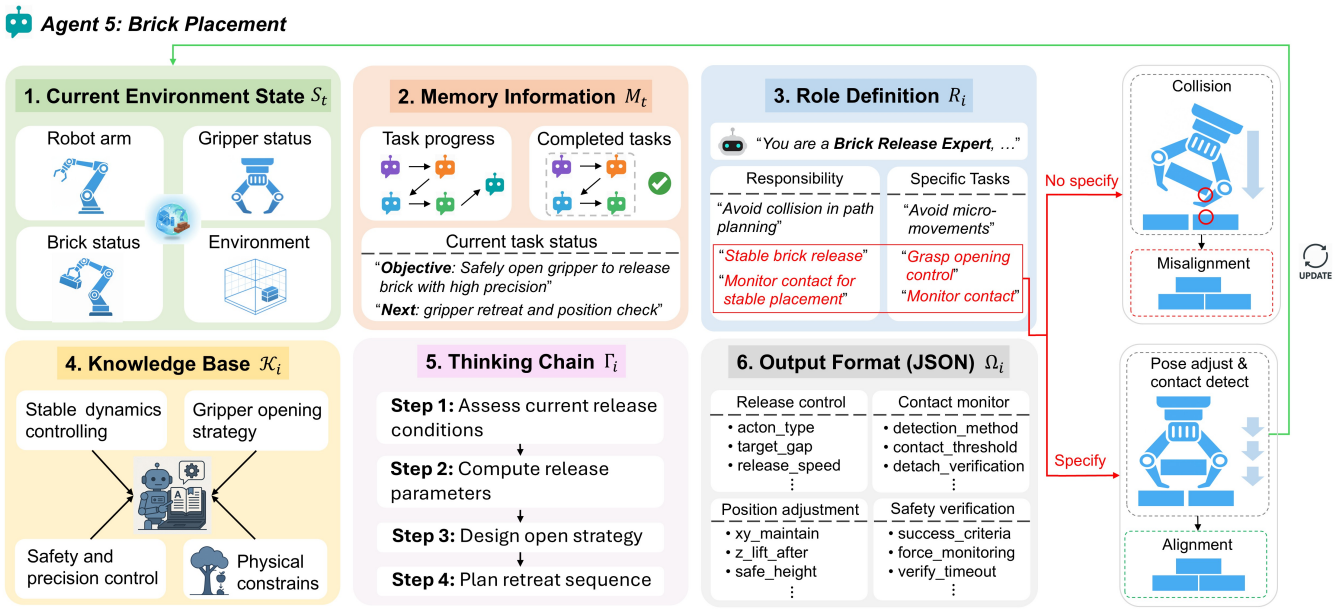

Fig. 3: Detailed architecture of Agent 5 (Brick Placement) with six prompt-driven components. (1) Current environment state: provides the latest world model; (2) Memory information: explains task progress of previous agents and the current task status; (3) Role definition: specifies the agent's function and responsibilities, including collision-avoidance tasks for stable brick placement. A comparison of the brick laying between specifying and not specifying is visualized aside; (4) Knowledge base: describes domain knowledge such as dynamics, gripper strategies, and safety constraints; (5) Thinking chain: outlines stepwise reasoning for placement and retreat; and (6) Output format: structured JSON commands for execution in the KUKA simulator.

d) Ag_4 Safe lift: The agent lifts vertically to height h_{safe} and evaluates slip risk. If risky, it returns to Ag_1 for re-grasp:

$$\sigma_4 = \mathbf{1}[\|v_{\text{brick}}\| \le v_{\text{th}} \wedge f_t \le \mu f_n], \qquad \sigma_4 = 0 \Rightarrow \text{goto } Ag_1. \tag{19}$$

e) Ag_5 Brick placement: The agent moves to the designated slot pose $p_{\text{slot}} \in \text{SE}(3)$, descends until contact, and checks alignment. If misaligned but in contact, raise by Δh and retry Ag_5:

$$\sigma_5 = \mathbf{1}[d_\perp \le \varepsilon_\perp \wedge \|\mathbf{e}_{xy}\| \le \varepsilon_{xy} \wedge e_\theta \le \varepsilon_\theta], \tag{20}$$

$$\sigma_5 = 0 \Rightarrow \text{raise by } \Delta h \text{ and repeat } Ag_5. \tag{21}$$

f) Ag_6 Return-to-ready: The agent retracts to a collision-free ready pose $p_{\text{ready}} \in \text{SE}(3)$ for the next brick:

$$\sigma_6 = \mathbf{1}\big[\text{CollisionFree}\big(\text{Path}(p_{\text{slot}} \to p_{\text{ready}})\big)\big]. \tag{22}$$

When all $\sigma_i = 1$, Ag_6 emits the next waypoint $w_{t+1} = p_{\text{ready}}$, which is passed to the controller. The loop in Section III-A.1 then updates perception S_{t+1} and repeats for the next brick.

This organization treats 3D action reasoning as a physics-guided inverse problem executed by a sequential gated LLM multi-agent pipeline. Waypoint control provides a safe and Markovian interface to the embedded controller. The operator F enforces physical constraints, while the LLM supplies structured proposals that exploit high-level priors. The proposed pipeline achieves using high-level tool invocation and prompting yet physically consistent manipulation for brick stacking.

B. Agent Construction via Structured Prompting

We instantiate one agent through a structured prompt that binds environment information, memory, role, tool knowledge, chain-of-thought guidance, and an explicit output schema. Fig. 3 gives one example for Ag_5 Brick placement. Formally, agent Ag_i at time t receives:

$$\text{Prompt}_i(t) \triangleq \langle S_t, M_t, R_i, \mathcal{K}_i, \Gamma_i, \Omega_i \rangle, \tag{23}$$

where $S_t \in \mathcal{S}$ is the current structured scene state, M_t is task memory, R_i is the role description, $\mathcal{K}_i$ is a set of callable knowledge/tools, Γ_i is the chain-of-thought scaffold, and Ω_i defines the output schema. Specifically, each module of the structured prompt can be described as follows.

- **Current environment state** S_t: robot and surroundings, including robot status, object poses/geometry, occupancy/free space, surface normals, and tolerance parameters.
- **Memory information** M_t: current task status, task progress and completed tasks, e.g., completed bricks, current brick index, current step for current brick, retry counters, step completion flags.
- **Role definition** R_i: concise instruction of the agent's responsibility and task specifications, including what to accomplish and how it interfaces with other agents.
- **Knowledge base** $\mathcal{K}_i$: basic knowledge in robotics, such as stable dynamics control method, safety and precision control strategies, gripper opening strategies, and physical constraint solving. These are memoried in callable

TABLE I: Comparison with classical baseline on two stacking patterns. Report rotation error (°), center offset (cm), and 3D box IoU (%). For rotation error and center offset, the lower the better. For 3D box IoU, the higher the better.

Method	Pyramid-like stacking			Grid-like stacking			Average		
	Rot. Err ↓	Ctr. Off. ↓	IoU ↑	Rot. Err ↓	Ctr. Off. ↓	IoU ↑	Rot. Err ↓	Ctr. Off. ↓	IoU ↑
Classical Controller (baseline)	1.103	4.318	38.51	0.939	4.379	37.72	1.004	4.314	38.38
ActionReasoning (Ours)	**0.583**	**0.561**	**89.03**	**0.822**	**0.712**	**87.02**	**0.703**	**0.637**	**88.03**

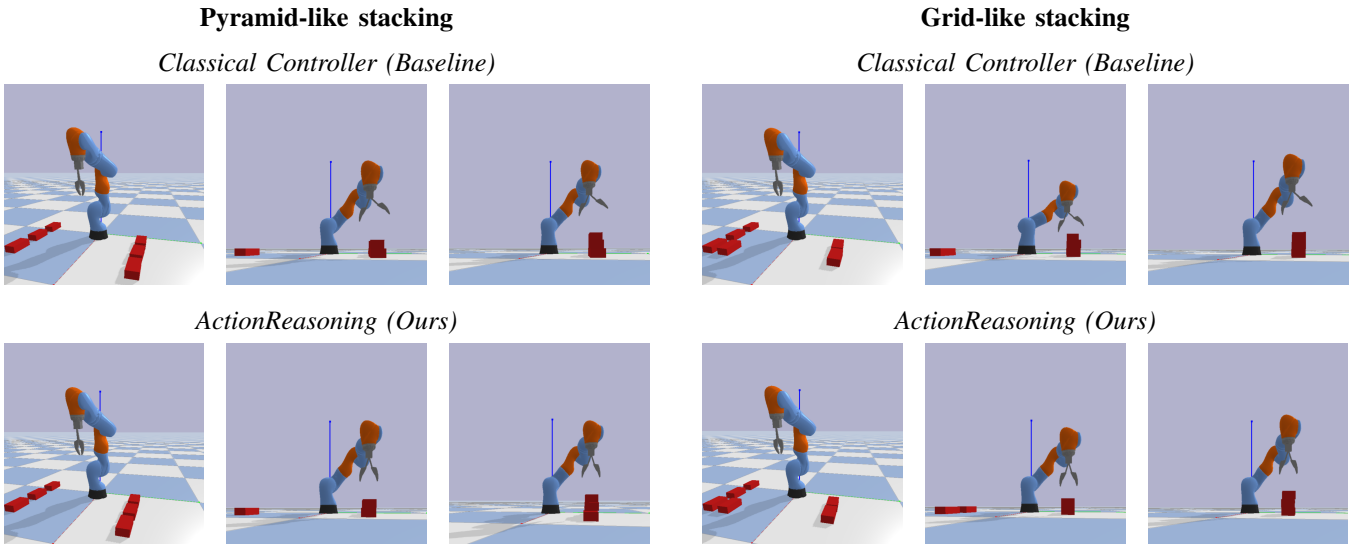

Fig. 4: Qualitative comparison across two stacking patterns. Each block shows the baseline (top row) and our method (bottom row). In each block of stacking pattern, columns show some stages of the placement cycle according to the timestamp from left to right. Our method achieves noticeably better brick alignment, as evidenced by the more neatly stacked bricks.

functions with typed I/O and usage (e.g., collision and contact checks, alignment estimators, safety margins).

- **Thinking Chain** $\Gamma_i = (\gamma_i^{(1)}, \ldots, \gamma_i^{(L_i)})$: a stepwise reasoning script describing how to think and decide.
- **Output Format** Ω_i: JSON-like fields for waypoints in SE(3) or code snippets to call tools.

The agent returns a tuple:

$$(m_i,\, y_i,\, \sigma_i,\, M_{t+1}) \;=\; Ag_i\big(\mathsf{Prompt}_i(t)\big), \tag{24}$$

where m_i is the agent's message, including intermediate rationale or parameters, $y_i \in \Omega_i$ is the actionable output, $\sigma_i \in \{0,1\}$ is an acceptance flag or gate, and M_{t+1} is the updated memory. In our setting,

$$y_i \in \Omega_i \;\subseteq\; \big(\mathrm{SE}(3) \;\cup\; \mathscr{C}\big),$$

i.e., either a waypoint $w \in \mathrm{SE}(3)$ for the manipulator or a small code snippet $c \in \mathscr{C}$ to invoke a tool. Typical tools in $\mathscr{K}_i$ include collision check, reachability, normal/contact force estimate, and planar translation and yaw errors. The gate σ_i is computed against explicit checks as Eq. (15).

IV. Experiments

A. Task and Setup

As illustrated in Fig. 1, the input to our system is a structured environment representation comprising: (i) a 3D scene state with brick poses/sizes/masses, current building progress, and placed-brick obstacles; (ii) the robot-arm state, including gripper openness and finger depth; and (iii) a target wall pattern G specifying the expected arrangement. Formally, at time t we observe $S_t \in \mathscr{S}$ and a fixed goal $G \in \mathscr{G}$. The proposed multi-agent planner in Section III-A produces executable waypoints $w_{t+1} \in \mathrm{SE}(3)$, and the embedded controller performs trajectory interpolation. After each action, the simulator updates the scene state to S_{t+1}, and planning proceeds to the next step.

To evaluate generality, we randomize the initial brick poses within the workspace $\mathscr{W} \subset \mathbb{R}^3$. We stack a total of 6 bricks to realize two types of patterns $\mathscr{G}$ as shown in the output part of Fig. 2: (1) a pyramid-like stacking, where each upper layer has fewer bricks centered on those below to form a triangular shape, with a gap of 0.05 m between adjacent bricks, (2) a grid-like stacking, where bricks are aligned in a regular rectangular grid with vertical and horizontal joints aligned, with a gap of 0.02 m between adjacent bricks. The operation is performed entirely in the physics-based simulator Pybullet [27]. Unless otherwise stated, contact/friction parameters and sensor noise follow the simulator defaults.

B. Evaluation Protocol and Metrics

We evaluate the mentioned two target wall patterns in the Section IV-A, each stacking $B = 6$ bricks per trial. For each pattern, we conduct $T_p = 10$ trials under randomized initial brick poses, yielding a total of $\sum_p T_p = 20$ trials. We report

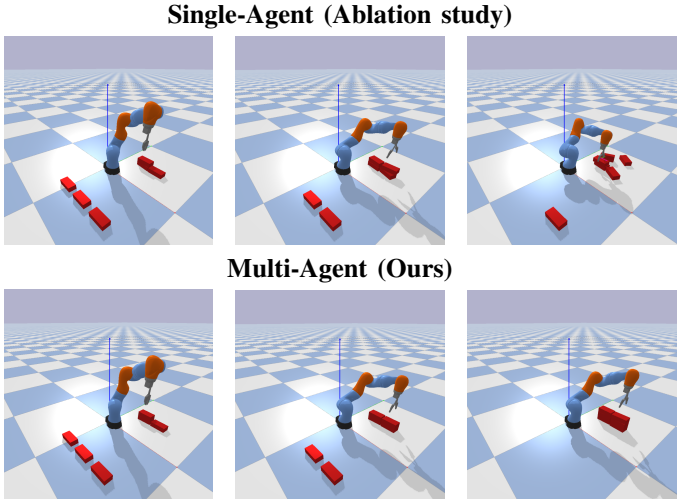

Fig. 5: Ablation visualizations for single agent. Rows depict single-agent vs multi-agent. Columns show key stages of the placement cycle according to the timestamp from left to right. The figure shows that the single agent toppled the structure and failed to complete the whole wall.

the alignment quality metrics over all 20 experiments.

1) Pose accuracy: For each successful trial r and brick b, the placed center and ground-truth center are denoted $\mathbf{c}_{r,b}$ and $\mathbf{c}^{\star}_{r,b}$, respectively. The center offset error is

$$e^{\mathrm{ctr}}_{r,b} \;=\; \left\|\mathbf{c}_{r,b}-\mathbf{c}^{\star}_{r,b}\right\|_2. \tag{25}$$

The rotation error (in degrees) uses the geodesic distance on $\mathrm{SO}(3)$ with rotation matrices $R_{r,b}, R^{\star}_{r,b}$:

$$e^{\mathrm{rot}}_{r,b} \;=\; \frac{180}{\pi}\,\arccos\!\left(\frac{\mathrm{trace}\big(R^{\star\top}_{r,b}R_{r,b}\big)-1}{2}\right). \tag{26}$$

2) 3D IoU: Let $B_{r,b}$ and $B^{\star}_{r,b}$ be the oriented 3D bounding boxes of the placed and ground-truth bricks, respectively. The 3D intersection-over-union (IoU) can be computed as:

$$\mathrm{IoU}_{r,b} \;=\; \frac{\mathrm{Vol}\big(B_{r,b}\cap B^{\star}_{r,b}\big)}{\mathrm{Vol}\big(B_{r,b}\cup B^{\star}_{r,b}\big)}. \tag{27}$$

Per-success trial averages are calculated over $B=6$ bricks:

$$\bar{e}^{\mathrm{ctr}}_{r}=\frac{1}{B}\sum_{b=1}^{B}e^{\mathrm{ctr}}_{r,b},\quad \bar{e}^{\mathrm{rot}}_{r}=\frac{1}{B}\sum_{b=1}^{B}e^{\mathrm{rot}}_{r,b},\quad \overline{\mathrm{IoU}}_{r}=\frac{1}{B}\sum_{b=1}^{B}\mathrm{IoU}_{r,b}. \tag{28}$$

Global means over all successful trials $\mathscr{R}$ are then:

$$\mathrm{Err}^{\mathrm{ctr}}=\frac{1}{|\mathscr{R}|}\sum_{r\in\mathscr{R}}\bar{e}^{\mathrm{ctr}}_{r},\quad \mathrm{Err}^{\mathrm{rot}}=\frac{1}{|\mathscr{R}|}\sum_{r\in\mathscr{R}}\bar{e}^{\mathrm{rot}}_{r},\quad \mathrm{IoU}=\frac{1}{|\mathscr{R}|}\sum_{r\in\mathscr{R}}\overline{\mathrm{IoU}}_{r}. \tag{29}$$

C. Baseline and Ablation Study Methods

All methods receive the same perception inputs and target brick stack pattern. All methods share an identical waypoint interface for low-level actuation. Thus, differences in outcomes stem from the reasoning layer, not from the low-level control stack.

1) Controller baseline: We compare our method, ActionReasoning, with a classical controller baseline that follows a fixed hand-scripted method. The baseline does not use thresholded contact events or environment collision checks. The reason is methodological: in our approach, these capabilities are implemented via LLM-driven tool calls, e.g., contact and collision queries, that do not require substantial domain-specific coding. However, the equivalent event handling in a traditional controller would require extensive bespoke code and expert tuning. To keep the engineering burden comparable across methods, we therefore exclude such event-based modules from the classical baseline and restrict it to a minimal hand-scripted pipeline. Therefore, our comparison reflects a realistic scenario in which an engineer with comparable expertise uses (i) a minimal scripted baseline or (ii) an LLM-driven reasoner that reduces low-level coding while increasing functional robustness through high-level tool calls and prompts.

2) Single-Agent ablation: To test whether multi-step agents are necessary, the single agent variant merges the prompts of all six agents into a single LLM call per waypoint, removing stage-wise gating.

D. Experiment Results

1) Controller baseline: Table I shows that our method markedly improves geometric accuracy over the classical controller. The mean rotation error drops from 1.004 cm to 0.703 cm (an 30.0% reduction), the mean center-offset drops from 4.314 cm to 0.637 cm (an 85.2% reduction), and the 3D box overlap increases from 0.3838 to 0.8803 (129% increase). The sequences in Fig. 4 corroborate the metrics: the baseline exhibits cumulative lateral drift and premature release during the place phase, leading to uneven stacks, whereas our method maintains consistent vertical descent, event-triggered release, and uniform brick spacing. Because both methods share the same perception inputs and low-level controller, these gains can be attributed to the proposed multi-agent physical reasoning, including thresholded contact detection and collision-gated releases as mentioned in Section IV-C.1.

2) Single-Agent ablation: We evaluate the single-agent variant under the wall-stacking setting. This ablation consistently underperforms the proposed staged pipeline, supporting the need for explicit role specialization and stage-wise gating. As shown in Fig. 5, the single-agent controller places the first four bricks with noticeably larger placement error and does not robustly handle the final two, frequently colliding with the structure and toppling the wall. We attribute this to the lack of inter-stage verification in the single-agent method, which allows early errors to propagate. This ablation underscores the necessity of multi-stage reasoning.

V. CONCLUSIONS

In this paper, we have introduced ActionReasoning, an LLM-driven approach to robotic manipulation that performs physical reasoning in $\mathrm{SE}(3)$. By conceptualizing robot control as a set of specialized agents with explicit roles and

gates, and by exposing LLMs to both an environment state S_t and callable physics/tool knowledge F, the system can reference relevant formulas and invoke verified functions to generate executable waypoints $w_{t+1} \in \mathrm{SE}(3)$. In the brick-laying domain, this design bridges the gap between understanding and doing: the LLM reasons over 3D scene geometry, object properties, and task constraints, while the controller handles trajectory interpolation and low-level servoing. Experiments in simulation demonstrate that the proposed multi-agent pipeline yields robust placement behavior, and ablations indicate that staged, role-specific reasoning is essential.

Future work. We will extend this framework beyond brick stacking to a broader set of complex unstructured construction site tasks and materials with varied physical properties: mortar deposition and leveling, block/stone/wood handling, fastening and drilling, compliant insertion, and cluttered-scene assembly with dynamic obstacles. Ultimately, our goal is a general-purpose construction robot capable of executing diverse tasks on-site and adapting to different objects, tools, and materials, so that a single robot can complete end-to-end building workflows with minimal low-level coding of specific task.

REFERENCES

[1] 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.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al., "Openvla: An open-source vision-language-action model," in Conference on Robot Learning. PMLR, 2025, pp. 2679–2713.

[3] Q. Vuong, S. Levine, H. R. Walke, K. Pertsch, A. Singh, R. Doshi, C. Xu, J. Luo, L. Tan, D. Shah, et al., "Open x-embodiment: Robotic learning datasets and rt-x models," in Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023, 2023.

[4] O. Mees, D. Ghosh, K. Pertsch, K. Black, H. R. Walke, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, et al., "Octo: An open-source generalist robot policy," in First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024, 2024.

[5] D. Ha and J. Schmidhuber, "World models," arXiv preprint arXiv:1803.10122, 2018. [Online]. Available: https://arxiv.org/abs/1803.10122

[6] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in Proceedings of the 36th International Conference on Machine Learning (ICML), ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 2555–2565. [Online]. Available: https://proceedings.mlr.press/v97/hafner19a.html

[7] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," arXiv preprint arXiv:1912.01603, 2020. [Online]. Available: https://arxiv.org/abs/1912.01603

[8] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, et al., "Genie: Generative interactive environments," in Forty-first International Conference on Machine Learning, 2024.

[9] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309–1332, 2016. [Online]. Available: https://rpg.ifi.uzh.ch/docs/TRO16_cadena.pdf

[10] S. Zhu, G. Wang, H. Blum, J. Liu, L. Song, M. Pollefeys, and H. Wang, "Sni-slam: Semantic neural implicit slam," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 21 167–21 177.

[11] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in Conference on Robot Learning. PMLR, 2025, pp. 4573–4602.

[12] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," Annual Review of Control, Robotics, and Autonomous Systems, vol. 4, pp. 265–293, 2021. [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-control-091420-084139

[13] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. Aparicio Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in IEEE International Conference on Robotics and Automation (ICRA), 2019, pp. 6023–6029.

[14] G. Wang, M. Xin, W. Wu, Z. Liu, and H. Wang, "Learning of long-horizon sparse-reward robotic manipulator tasks with base controllers," IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 3, pp. 4072–4081, 2022.

[15] Y. Wu, L. Pan, W. Wu, G. Wang, Y. Miao, F. Xu, and H. Wang, "Rl-gsbridge: 3d gaussian splatting based real2sim2real method for robotic manipulation learning," in 2025 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2025, pp. 192–198.

[16] A. Brohan, N. Brown, J. Carbajal, et al., "Rt-1: Robotics transformer for real-world control at scale," arXiv preprint arXiv:2212.06817, 2022. [Online]. Available: https://arxiv.org/abs/2212.06817

[17] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al., "Bridgedata v2: A dataset for robot learning at scale," in Conference on Robot Learning. PMLR, 2023, pp. 1723–1736.

[18] T. Z. Zhao et al., "Learning fine-grained bimanual manipulation with low-cost open-source teleoperation," arXiv preprint arXiv:2304.13705, 2023. [Online]. Available: https://arxiv.org/abs/2304.13705

[19] J. Aldaco, T. Armstrong, J. Bingham, et al., "Aloha 2: An enhanced low-cost hardware for bimanual teleoperation," arXiv preprint arXiv:2405.02292, 2024. [Online]. Available: https://arxiv.org/abs/2405.02292

[20] 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.

[21] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," Advances in neural information processing systems, vol. 36, pp. 11 809–11 822, 2023.

[22] M. Pan, J. Zhang, T. Wu, Y. Zhao, W. Gao, and H. Dong, "Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints," in Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, pp. 17 359–17 369.

[23] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "Camel: Communicative agents for" mind" exploration of large language model society," Advances in Neural Information Processing Systems, vol. 36, pp. 51 991–52 008, 2023.

[24] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al., "Autogen: Enabling next-gen llm applications via multi-agent conversations," in First Conference on Language Modeling, 2024.

[25] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," arXiv preprint arXiv:2305.16291, 2023. [Online]. Available: https://arxiv.org/abs/2305.16291

[26] A. Ghafarollahi and M. J. Buehler, "Automating alloy design and discovery with physics-aware multimodal multiagent ai," Proceedings of the National Academy of Sciences, vol. 122, no. 4, p. e2414074122, 2025.

[27] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning.(2016)," URL http://pybullet. org, 2016.