WCM: World-Cognition Model for Generalizable Human–Robot Interaction

Yuzhen Chen[1], K. Zhou[2]
[1]Harvard University [2]Massachusetts Institute of Technology

***Abstract*—Language agents can now interact fluently with users in software, but robots still struggle to bring comparable interaction to physical tasks. Current robot-control paradigms, including vision-language-action policies and world-model-based planners, are mainly optimized for instruction execution, leaving users with little visibility into why an action is chosen and few mechanisms to redirect, correct, or teach the robot through interaction. We present the World-Cognition Model (WCM), a human-centered embodied agent built on the SLAK architecture (Sensing, Logic, Action, and Knowledge) and an asynchronous runtime. SLAK separates perception, reasoning, control, and memory, while the runtime allows reasoning, dialogue, and execution to proceed concurrently. WCM further introduces a human-in-the-loop teaching mode that enables users to interactively teach the robot difficult or long-horizon tasks. Teaching episodes and autonomous task rollouts are refined into chain-of-thought supervision to continually improve the model. WCM achieves a 73.8% average success rate across nine real-world human–robot interaction tasks, including tasks held out from CoT fine-tuning and a long-horizon task learned through teaching.**

I. Introduction

Language has become a common interface for robot control, but following language instructions is not the same as human–robot interaction. In software, language agents allow users to refine goals, ask for explanations, and steer a task as it unfolds. In embodied systems, however, language is still often treated as an input condition for action generation rather than a persistent channel for interaction, explanation, and feedback-driven learning. As a result, even robots that can execute language instructions offer limited support for mid-execution redirection, human-readable explanations, or learning from user feedback.

Vision-language-action (VLA) policies [3, 16, 2] and world-model-based planners [12, 11, 4, 27] expose this gap in different ways. VLA policies can accept language commands, but language usually serves as a conditioning signal for action prediction rather than a persistent channel through which a user can intervene, revise the goal, or teach the robot during execution. Their decisions are also difficult to inspect: even when a pretrained vision-language backbone provides semantic understanding, the final motion is produced by an action decoder or policy, leaving the link between high-level reasoning and physical behavior implicit. World-model-based planners face a related limitation. They reason through predicted future states or imagined rollouts, which can support planning, but these rollouts are not by themselves human-readable explanations, nor do they naturally provide a mechanism for real-time conversational correction. Thus, both families advance language-conditioned execution, but neither makes interaction, legibility, and feedback-driven learning a first-class part of the control loop.

Interactive robots therefore require more than a stronger language-conditioned policy: perception, reasoning, action, memory, and human feedback must remain connected throughout execution. The robot should ground a user's instruction in the current scene, expose the reason behind its next action, revise that action when the user intervenes, and turn interaction itself into future training signal [7, 13].

We instantiate this view in the World-Cognition Model (WCM), a human-centered embodied agent built on the SLAK architecture and an asynchronous runtime. SLAK separates Sensing, Logic, Action, and Knowledge, making perception, reasoning, control, and memory explicit rather than hidden inside a single opaque policy. The runtime allows reasoning, dialogue, and execution to proceed concurrently, while WCM converts autonomous rollouts and human teaching episodes into chain-of-thought supervision for continual improvement [25, 14].

We evaluate WCM on a low-cost functional mobile-manipulation platform and propose nine real-world human–robot interaction tasks covering object retrieval, handover, tool use, drawer manipulation, and trash disposal. WCM achieves a 73.8% average success rate, including tasks held out from CoT fine-tuning and a long-horizon task learned through teaching.

Our contributions are threefold. **First,** we introduce WCM, an interactive embodied system with explicit SLAK layers and asynchronous execution. **Second,** we propose a human-in-the-loop teaching and CoT distillation pipeline that turns autonomous rollouts and teaching episodes into training signal. **Third,** we build a low-cost mobile-manipulation platform and introduce a nine-task real-world HRI suite for evaluating language-guided interaction, real-time correction, and task learning.

II. Method: The World-Cognition Model

A. System Architecture and Runtime

1) SLAK Architecture: SLAK is the core system architecture of WCM, consisting of four explicit layers: Sensing, Logic, Action, and Knowledge (Fig. 1). These layers separate scene understanding, reasoning, robot control, and memory while communicating through explicit state exchange. This design lets WCM update the scene, goal, memory, and action outcomes during execution, rather than hiding the full

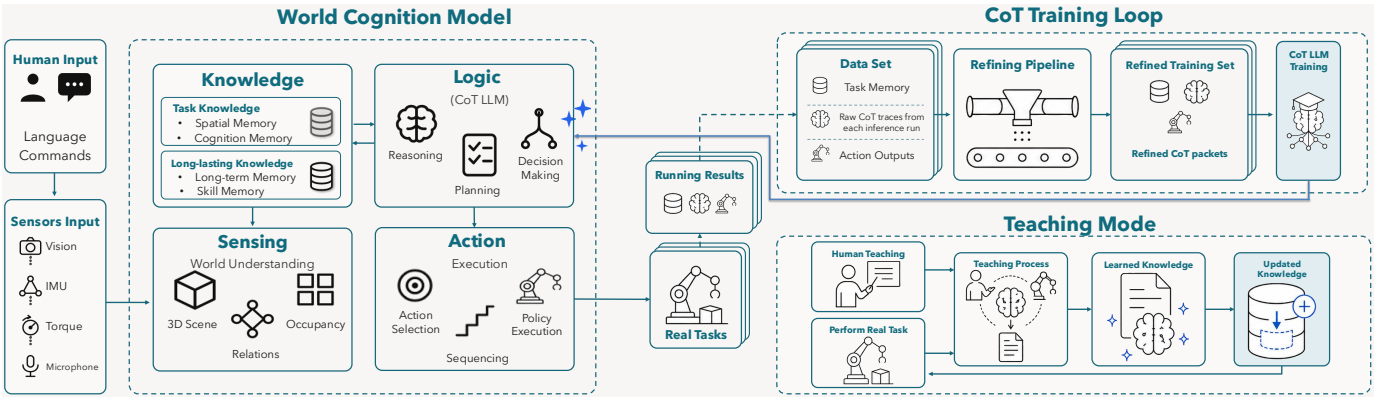

Fig. 1. **The SLAK architecture.** WCM organizes sensing, logic, action, and knowledge as explicit layers connected to an interaction-driven learning loop. Autonomous executions and human teaching episodes are refined into CoT training data for continual improvement.

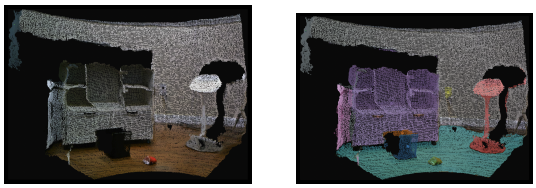

Fig. 2. **Sensing layer output.** Raw fused sensor view (left) and interaction-aware scene state (right), with local geometry, semantic cues, and part-level interaction regions.

perception-to-action process inside a single end-to-end policy [3, 16, 2].

a) Sensing layer: The Sensing layer converts multimodal sensor streams into an interaction-aware scene state. It integrates vision, depth, audio, IMU, and force/torque signals to maintain local 3D structure, object relations, occupancy, and semantic cues around the robot. Unlike streaming 3D perception methods that primarily reconstruct geometry from image sequences [17, 26], or open-vocabulary grounding and segmentation models that localize language-specified concepts as boxes or masks [20, 22, 5], WCM estimates the scene properties needed for robot interaction. These include pixel- and part-level cues for grasping, opening, placement, and disposal, as well as coarse physical and material attributes such as rigidity, fragility, and grasp suitability [6]. The resulting state informs Logic about available local interactions and the physical constraints that should shape the next action.

b) Logic layer: The Logic layer is WCM's reasoning, planning, and decision-making module. It conditions each decision on the user's current instruction, the structured scene state from Sensing, and the relevant memory retrieved from Knowledge.

Rather than committing to a complete task script upfront, Logic follows an incremental plan–reason–validate loop. At step t, Logic forms a decision context $x_t = (s_t, u_t, m_t)$ from the current scene state, user instruction, and retrieved memory, and produces a reasoning trace, the next action, and its expected outcome:

$$(r_t, a_t, \hat{o}_t) = f_\theta(x_t), \qquad v_t = \begin{cases} 1, & \Delta(o_t, \hat{o}_t) \le \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where f_θ is the Logic model, θ denotes its parameters, o_t is the realized outcome returned after execution, Δ measures the mismatch between expected and realized outcomes, ϵ is a validation threshold, and $\mathbf{1}[\cdot]$ is the indicator function, which returns 1 when the condition holds and 0 otherwise. Thus, v_t is a binary validation flag: if $v_t = 1$, the task proceeds; otherwise Logic re-reasons from the updated state or asks the user for clarification.

This stepwise structure keeps decisions grounded in the current scene, exposes the robot's intent and expected outcome in natural language, and limits error propagation when the environment changes or the user redirects the task [1, 10, 15].

c) Action layer: The Action layer converts Logic's selected step into executable robot behavior. Given a step and its expected outcome, Action binds it to the appropriate low-level skill or policy, sequences the required motion commands, and executes them on the robot hardware. These skills cover navigation, reaching, grasping, placing, handover, and simple articulated-object interactions, and are reused across tasks rather than trained as task-specific policies. During execution, Action reports status, failures, and realized outcomes back to Logic and Knowledge for validation and replanning when necessary. This separation keeps high-level reasoning independent from the underlying controller, allowing the action library or hardware stack to be improved without changing the reasoning interface [18, 23].

d) Knowledge layer: The Knowledge layer stores and updates the state and experience WCM accumulates during task execution and long-term use. It maintains short-term task memory, including the current goal, observed objects, spatial context, recent actions, outcomes, and user feedback, as well as longer-term knowledge such as task experience, object-interaction history, and skill memory. This memory allows WCM to preserve context across a continuous interaction rather than treating each decision as an isolated response to the current observation.

During execution, Sensing, Logic, and Action write scene changes, active goals, reasoning states, action outcomes, failures, and user feedback into Knowledge as a shared task state. Logic retrieves this state when deciding the next step, keeping

execution, interaction feedback, and later learning connected within the same system loop.

Taken together, SLAK makes WCM's execution state explicit across sensing, reasoning, action, and memory. This explicit state interface provides the basis for the asynchronous runtime, which coordinates concurrent reasoning, execution, and state updates while preserving a shared task context.

2) Asynchronous Runtime: Building on SLAK's explicit state interface, the asynchronous runtime coordinates reasoning, execution, and state updates as concurrent loops. A conventional robot-control loop runs perception, reasoning, action, and feedback sequentially, so model inference can block execution and delay responses to new user commands. WCM instead runs a reasoning loop that proposes actions, an execution loop that commits actions to hardware, and a state-update loop that incorporates new observations, outcomes, and user feedback into the shared task state.

Asynchrony does not mean that WCM blindly follows an outdated action queue. Before committing queued actions and after execution, the runtime applies the validation check in Eq. (1) using the latest scene state, user input, and realized outcome. If the context no longer matches the expectation, stale actions are discarded and Logic re-reasons from the updated state. This decouples reasoning from execution while preserving responsiveness to dynamic environments, allowing the robot to move, explain, listen for corrections, and update the task plan as the situation changes.

B. Interaction-Driven Learning

Beyond interactive execution, WCM turns real task execution into a source of continual learning. Each task episode is decomposed into step-level decision records rather than stored as a teleoperated motor trajectory. For a task episode of length T, we write

$$e = \{d_t\}_{t=1}^{T}, \quad d_t = (s_t, u_t, m_t, r_t, a_t, \hat{o}_t, o_t), \tag{2}$$

where s_t is the scene state, u_t the user goal or instruction, m_t the retrieved memory, r_t the Logic reasoning, a_t the selected action, $\hat{o}_t$ the expected outcome, and o_t the realized outcome.

These records come from two sources: autonomous executions, including both successes and failures, and human-guided teaching episodes for difficult or long-horizon tasks. Let $\mathcal{D}_{\text{auto}}$ and $\mathcal{D}_{\text{teach}}$ denote the corresponding decision records. The offline refinement pipeline produces refined CoT examples $\mathcal{D}'_{\text{auto}} = \mathcal{R}(\mathcal{D}_{\text{auto}})$ and $\mathcal{D}'_{\text{teach}} = \mathcal{R}(\mathcal{D}_{\text{teach}})$, where $\mathcal{R}$ denotes the refinement procedure. Because teaching episodes contain direct human guidance and clearer task decomposition, we weight them more heavily during distillation. Let $\mathcal{L}_{\text{auto}}(\theta)$ and $\mathcal{L}_{\text{teach}}(\theta)$ denote the CoT negative log-likelihood losses over $\mathcal{D}'_{\text{auto}}$ and $\mathcal{D}'_{\text{teach}}$, respectively. The weighted distillation objective is

$$\mathcal{L}_{\text{distill}}(\theta) = \mathcal{L}_{\text{auto}}(\theta) + \lambda \mathcal{L}_{\text{teach}}(\theta), \tag{3}$$

where $\lambda \geq 1$ controls the relative weight of human-guided teaching examples. WCM minimizes this objective to update the reasoning model.

1) Autonomous Data Flywheel: During autonomous execution, WCM records the reasoning-action process behind each completed or failed task. Successful episodes provide examples of effective task decomposition, grounding, action selection, and outcome validation. Failed episodes are equally useful: they reveal where the model misunderstood the scene, selected an ineffective action, predicted the wrong outcome, or recovered through unnecessary trial and error. Unlike VLA-style training on teleoperated state-action trajectories [3, 16, 2], WCM uses the decision records in Eq. (2) as step-level reasoning supervision rather than motor-action supervision.

Raw execution records are not used directly for training. Because real robot runs can contain redundant reasoning, repeated attempts, or ineffective intermediate actions, WCM sends raw CoT traces to an offline refinement pipeline. A stronger model ensemble reviews the recorded reasoning and outcomes, removes unnecessary detours, and produces cleaner reasoning traces through voting or consensus. The result preserves the causal structure of real execution while turning noisy behavior into higher-quality CoT supervision [25, 21, 19, 24].

The refined autonomous examples are distilled under the shared objective in Eq. (3), closing the autonomous data flywheel: execution produces decision records, refinement improves the supervision, and distillation improves later executions. Over time, WCM accumulates reusable reasoning patterns for object interaction, action sequencing, outcome prediction, and failure recovery, rather than memorizing a single task script.

2) Human-in-the-Loop Teaching Mode: The autonomous data flywheel improves WCM from its own successes and failures, but some unseen or long-horizon tasks remain inefficient to discover through trial and error alone. For such cases, WCM introduces a human-in-the-loop teaching mode. Rather than requiring low-level teleoperation, the user teaches through natural language, guiding the robot step by step by indicating what to attend to, which object or part to act on, and when to advance to the next stage.

During teaching, WCM records the same step-level decision format as autonomous execution, augmented with explicit human guidance and feedback. These examples are often more informative than unguided rollouts because they provide clearer task decomposition, direct correction signals, and human choices about how a difficult task should be staged. This makes them especially useful for long-horizon or previously unseen tasks, where autonomous exploration alone may be slow or brittle.

Teaching episodes enter the same refinement pipeline and shared distillation objective as autonomous execution records, but are weighted by λ in Eq. (3) because they provide direct human guidance. In the short term, taught procedures are stored in the Knowledge layer for reuse on similar tasks. In the long term, refined teaching examples are distilled into the reasoning model, allowing the robot to internalize human guidance as reusable task competence [25, 14, 8, 9].

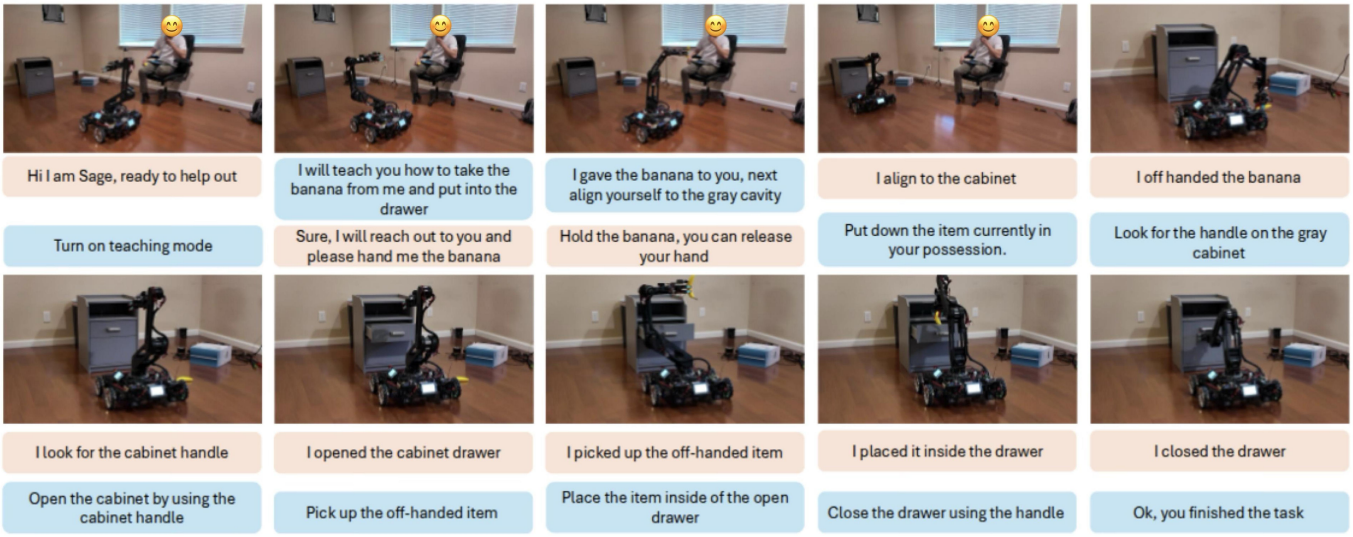

Fig. 3. **Human-in-the-loop teaching.** A user guides a long-horizon task through natural language; the episode is later refined and distilled with autonomous records.

III. Experiments

We evaluate WCM on a low-cost mobile-manipulation platform and a nine-task real-world HRI suite, measuring task success, held-out transfer, teaching-based skill acquisition, system efficiency, and ablated components.

A. Hardware

WCM runs on a low-cost mobile-manipulation platform: a Mecanum-wheel base, a 5-DOF arm ($\sim$1.3 m reach), and a 640×480 RGB-D camera. The complete robot costs under \$2,000, while reasoning runs off-board on a shared NVIDIA RTX 5090 over the local network; we therefore report robot hardware cost separately from compute.

B. Nine-Task Human–Robot Interaction Suite

We propose a nine-task real-world human–robot interaction suite to evaluate WCM beyond isolated manipulation primitives. Each task begins with a spoken instruction and requires language understanding, scene grounding, and physical execution. The suite covers object retrieval, handover, tool use, drawer manipulation, and trash disposal. Per-task success rates, trials, and run times are reported in Table I, with frame-by-frame demonstrations in Figs. 4–12 (Appendix A).

The robot communicates in real time during these tasks: it listens, confirms, asks, and explains while acting. Four tasks are held out from CoT fine-tuning yet still succeed, demonstrating transfer beyond the training tasks. The hardest task, "Take the screwdriver to the drawer," cannot be solved reliably before teaching; after interactive teaching it reaches 69% success, and after distillation it rises to 82%, testing the full teaching-to-distillation loop.

C. Case-Study Comparison

We present a case-study comparison of WCM with X-Square's WALL-OSS, reproducing its task suite as closely as our platform allows. Because the two robots differ in body, kinematics, sensors, and execution stack, this comparison is not a controlled benchmark and should not be interpreted as a definitive ranking. Instead, it illustrates system-level properties such as task speed, intermittent GPU use, and success on comparable tasks. Per-task speed, GPU occupancy, and success rates appear in Appendix B (Table II).

D. Ablations

Finally, we ablate the asynchronous runtime and the interaction-aware Sensing layer on a shared five-task subset. Removing asynchrony forces the robot to wait between reasoning and action, increasing end-to-end run time by 1.4–$1.7\times$. Replacing Sensing with a plain VLM and 2D grounding sharply reduces success on part-level interaction tasks; Screwdriver-to-Drawer falls from 69% to 0%. Results are reported in Appendix C (Table III).

IV. Conclusion

We present WCM, a human-centered embodied agent that combines the SLAK architecture, an asynchronous runtime, and interaction-driven learning for generalizable human–robot interaction. WCM allows the robot to maintain task context, explain its intent, respond to mid-execution corrections, and improve from both autonomous execution and human teaching. On a low-cost mobile-manipulation platform, WCM achieves a 73.8% average success rate across nine real-world interaction tasks, including held-out tasks and a long-horizon task learned through teaching. While the current system remains limited by its hardware and evaluated through a case-study comparison, it points toward robots that can act, explain, be corrected, and improve with people in the loop.

Acknowledgments

We thank T. Lihe. and Andrew Hoang for their contributions to the system implementation and experiments. This work was funded by Cyberbrain, Inc. as part of its internal R&D program; the system, datasets, and related IP are owned by Cyberbrain, Inc.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022. arXiv:2204.01691.

[2] Kevin Black, Noah Brown, Danny Driess, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023. arXiv:2307.15818.

[4] Jake Bruce, Michael Dennis, Ashley Edwards, et al. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.15391.

[5] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[6] Yuzhen Chen, Hojun Son, and Arpan Kusari. Matpredict: A dataset and benchmark for learning material properties of diverse indoor objects. *arXiv preprint arXiv:2505.13201*, 2025.

[7] Herbert H. Clark and Susan E. Brennan. Grounding in communication. In Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley, editors, *Perspectives on Socially Shared Cognition*, pages 127–149. American Psychological Association, 1991.

[8] Yuntian Deng, Kiran Prasad, Roland Fernandez, et al. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

[9] Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

[10] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, et al. PaLM-E: An embodied multimodal language model. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2303.03378.

[11] Yilun Du, Mengjiao Yang, Bo Dai, et al. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2302.00111.

[12] David Ha and Jürgen Schmidhuber. World models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. arXiv:1803.10122.

[13] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 159–166, 1999.

[14] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, et al. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics (ACL)*, 2023. arXiv:2305.02301.

[15] Wenlong Huang, Fei Xia, Ted Xiao, et al. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning (CoRL)*, 2022. arXiv:2207.05608.

[16] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024. arXiv:2406.09246.

[17] Yushi Lan, Yihang Luo, Fangzhou Hong, Shangchen Zhou, Honghua Chen, Zhaoyang Lyu, Shuai Yang, Bo Dai, Chen Change Loy, and Xingang Pan. Stream3r: Scalable sequential 3d reconstruction with causal transformer. *arXiv preprint arXiv:2508.10893*, 2025.

[18] Jacky Liang, Wenlong Huang, Fei Xia, et al. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. arXiv:2209.07753.

[19] Hunter Lightman, Vineet Kosaraju, Yura Burda, et al. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.20050.

[20] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, 2024.

[21] Maxwell Nye, Anders Andreassen, Guy Gur-Ari, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[22] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[23] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

[24] Jonathan Uesato, Nate Kushman, Ramana Kumar, et al. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

[25] Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. Chain-of-thought prompting elicits reasoning in large language

models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2201.11903.

[26] Kaichen Zhou, Yuhan Wang, Grace Chen, Xinhai Chang, Gaspard Beaudouin, Fangneng Zhan, Paul Pu Liang, and Mengyu Wang. Page-4d: Disentangled pose and geometry estimation for 4d perception. *arXiv preprint arXiv:2510.17568*, 2025.

[27] Kaichen Zhou, Yuzhen Chen, Fangneng Zhan, Hang Hua, Grace Chen, Xinhai Chang, Ao Qu, Yilun Du, Zhuang Liu, Paul Pu Liang, et al. Gem-4d: Geometry-enhanced video world models for robot manipulation. *arXiv preprint arXiv:2605.22882*, 2026.

APPENDIX A
HUMAN–ROBOT INTERACTION DEMOS

TABLE I

HUMAN–ROBOT INTERACTION DEMOS. PER-TASK SUCCESS RATE (%) OF WCM ON NINE LIVE INTERACTION TASKS, EACH COUPLING DIALOGUE WITH PHYSICAL HANDOVER TO A PERSON. TASK LABELS ARE THE SPOKEN INSTRUCTIONS OF THE MATCHING DEMO FIGURES (FIGS. 4–12). BECAUSE CONVENTIONAL VLA POLICIES TYPICALLY DO NOT TREAT CONTINUOUS DIALOGUE, INTERPRETABLE INTERRUPTION, AND TEACHING MODE AS CORE CLOSED-LOOP CAPABILITIES, WE REPORT WCM ONLY.

Spoken instruction	**SR (%)**	**Trials**	**Time (s)**
"Grab me the popcorn."	85	11/13	72
"Hand me the honey."	79	11/14	34
"Hand me the jacket."	67	6/9	54
"Hand me the green ratchet."	64	7/11	36
"Hold the BBQ tongs."	86	12/14	48
"Grab me the water bottle."	83[1]	10/12	30
"Take the screwdriver to the drawer."	69[2]	9/13	165
"Put the cup in the trash can." (Outdoor)	67[1]	10/15	46
"Put the bottle in the trash can."	64[1]	9/14	34

[1] Held out from the CoT training set; solved zero-shot by transfer. [2] Held out and acquired through teaching mode: the reported 69% is post-teaching but pre-distillation (below 20% before teaching, 82% after CoT distillation). **Trials** reports successful / total attempts; **SR** is their ratio. **Time** is a typical end-to-end run time (seconds), from robot-motion onset to task completion; values are approximate.

Figures 4–12 show representative frame sequences from the live demos summarized in Table I, each capturing the dialogue-driven handover from the robot's first look at the scene through grasp and delivery to the person.

Fig. 4. **"Grab me the popcorn."** Object retrieval and handover.

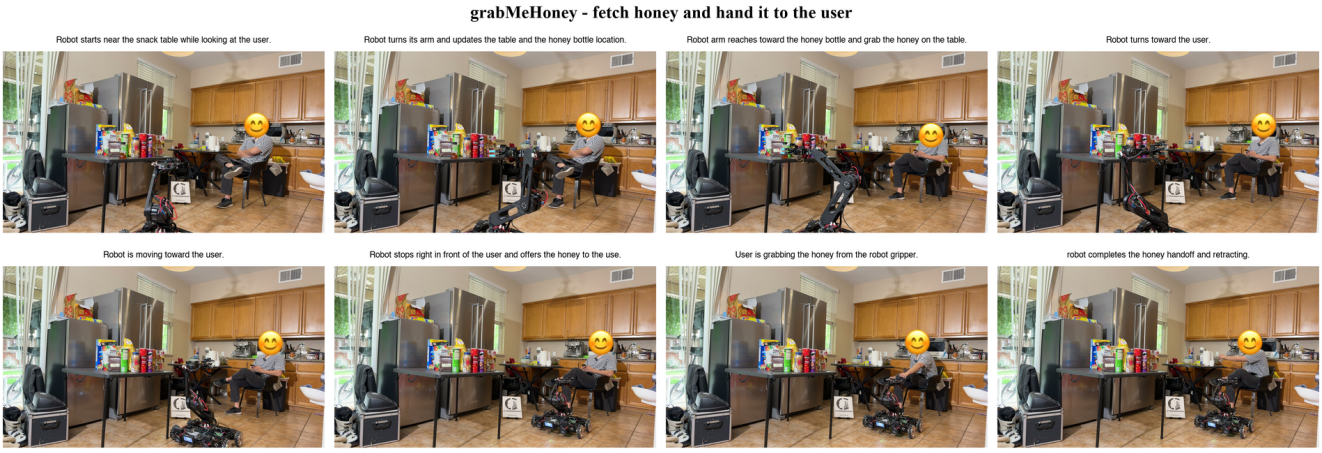

Fig. 5. **"Hand me the honey."** The robot locates the honey, grasps it, and hands it to the person.

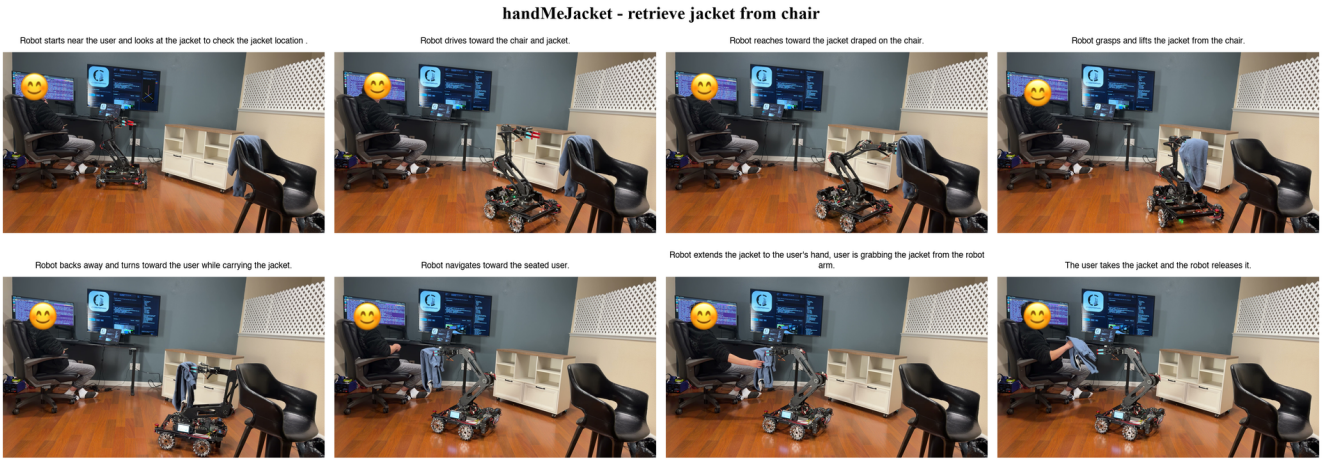

Fig. 6. **"Hand me the jacket."** Grasping a deformable object and delivering it to the person.

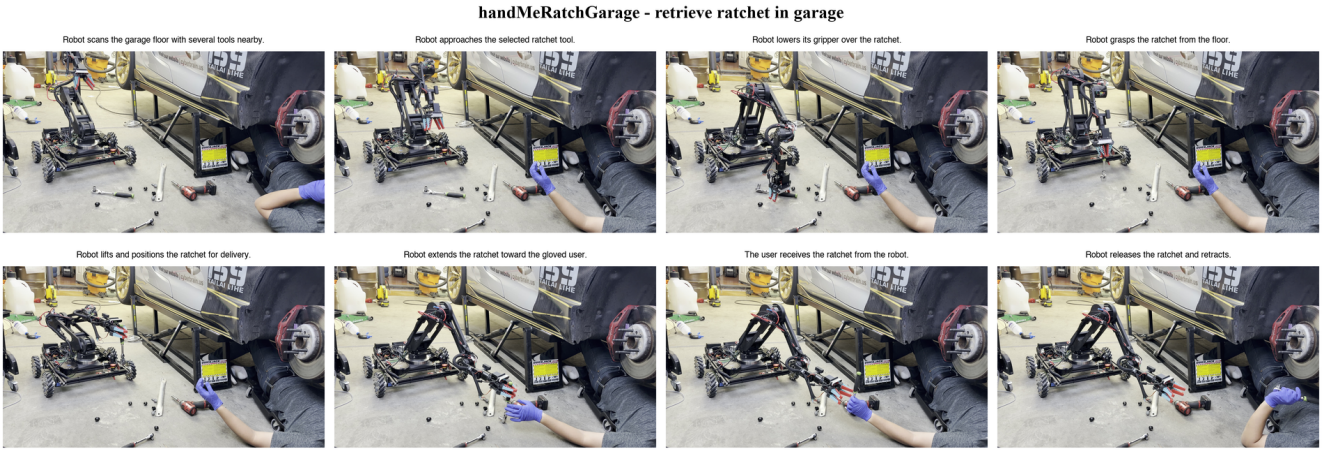

Fig. 7. **"Hand me the green ratchet."** Retrieving a tool in a garage setting.

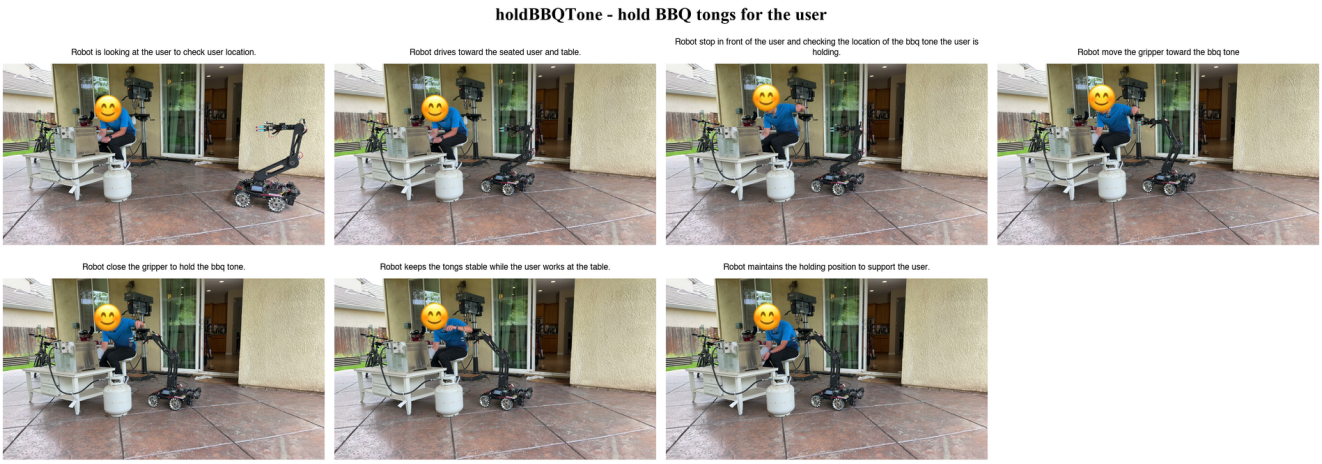

Fig. 8. **"Hold the BBQ tongs."** Grasping and presenting the tongs to the person.

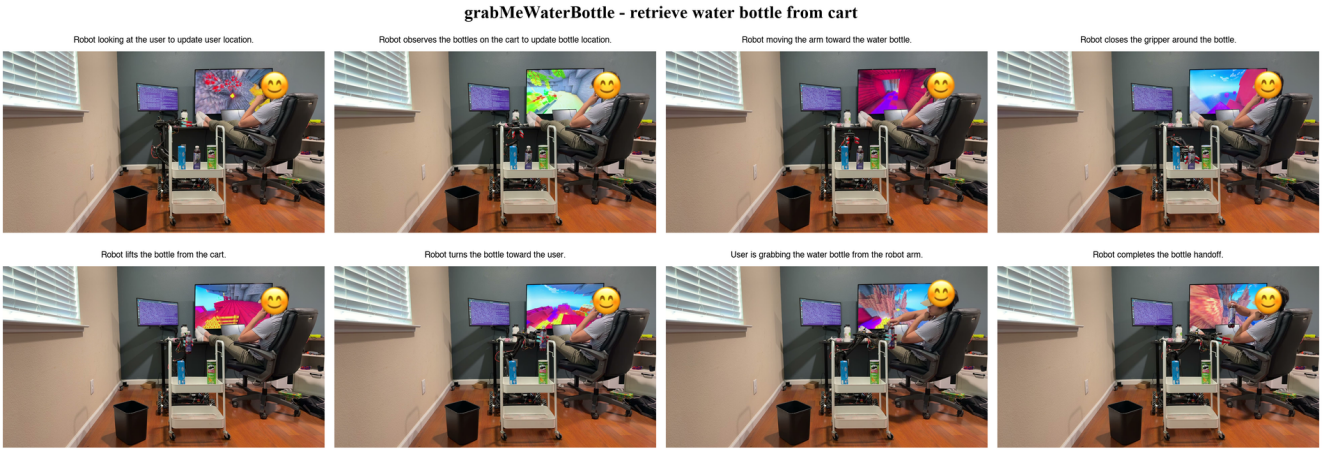

Fig. 9. **"Grab me the water bottle."** Object retrieval and handover.

Fig. 10. **"Take the screwdriver to the drawer."** A two-stage manipulation: open the drawer, then place the screwdriver inside.

Fig. 11. **"Put the cup in the trash can."** (Outdoor) A multi-step task: pick up the cup, navigate outside, and dispose of it.

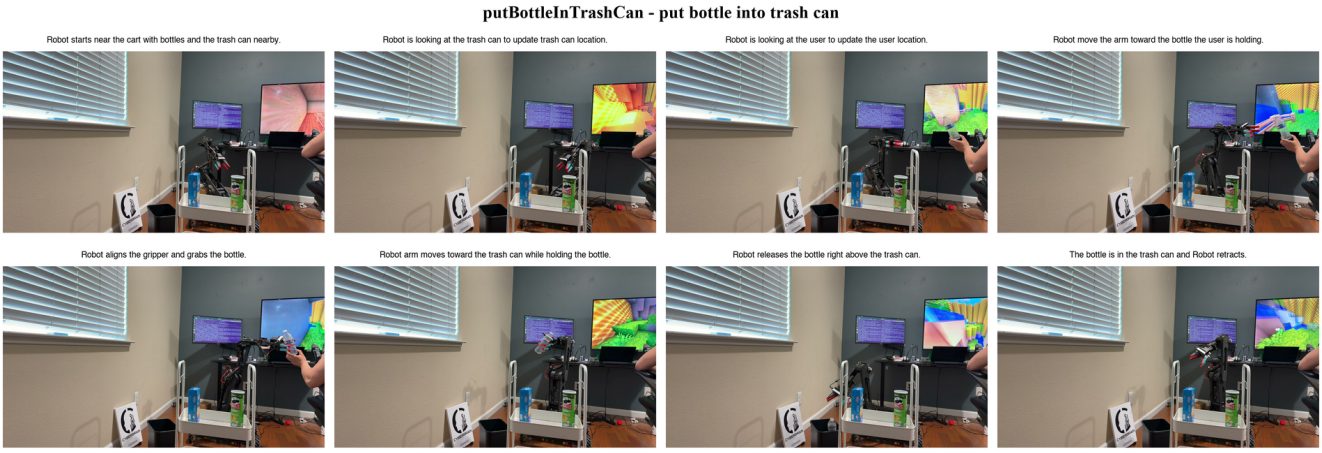

Fig. 12. **"Put the bottle in the trash can."** A multi-step task: pick up the bottle, locate the trash can, and dispose of it.

APPENDIX B
PERFORMANCE COMPARISON RESULTS

TABLE II

CASE-STUDY COMPARISON. PER-TASK COMPARISON OF **WCM (OURS)** AGAINST **X-SQ.** (X-SQUARE ROBOT; MODEL: **WALL-OSS**) ON SIX TASKS ACROSS THREE METRICS. ARROWS GIVE THE BETTER DIRECTION; **BOLD** MARKS THE BETTER METHOD PER COMPARED CELL.

	Movement time (s) ↓		GPU busy time (s) ↓		Success (%) ↑
Task	WCM	X-Sq.	WCM	X-Sq. (est.)	WCM
ColorPlateSort	**25.0**	109.0	**19.4**	109.0	92
LabelRecognition	**36.0**	116.0	**18.4**	116.0	90
PickUpFruits	**16.6**	18.5	**6.4**	18.5	90
ThrowTrash	**80.5**	258.0	**22.5**	258.0	86
PickUpCube	**16.1**	43.2	**5.6**	43.2	92
SolveMathProblem	**57.0**	76.0	**59.9**	76.0	78
Average	**38.5**	103.5	**22.0**	103.5	88.0

Movement time is measured from the onset of robot motion to completion of the target task, matching the reference videos of the baseline (which start when the robot begins to move). **GPU busy time** is the wall-clock time the GPU spends computing for the task. A VLA runs on the GPU at every control step, so for X-Square we estimate its GPU busy time as the full movement time (continuous occupancy); WCM uses the GPU only intermittently, freeing it for other robots. WCM also spends 3–5 s on a post-task summary, so its GPU stays busy slightly past the end of motion, which is why GPU busy time can exceed movement time (e.g., SolveMathProblem, 59.9 vs 57.0 s). **Success.** X-Square did not provide per-task success rates, so the success column reports WCM only.

APPENDIX C
ABLATIONS

TABLE III

ABLATIONS. REMOVING THE **ASYNCHRONOUS RUNTIME** (THE ROBOT MUST HALT TO THINK BETWEEN ACTIONS) INFLATES END-TO-END RUN TIME; REPLACING **INTERACTION-AWARE SENSING LAYER** WITH A PLAIN VLM AND 2D GROUNDING (NO PART-LEVEL INTERACTION CUES) REDUCES THE SUCCESS RATE. BOTH ARE EVALUATED ON THE SAME FIVE TASKS. "WCM" IS THE FULL SYSTEM—ITS TIMES AND SUCCESS RATES MATCH TABLE I; ARROWS GIVE THE BETTER DIRECTION AND **BOLD** MARKS THE FULL SYSTEM.

	Time (s) ↓		Success (%) ↑	
Task	WCM	w/o Async	WCM	w/o Sensing
"Hand me the green ratchet."	**36**	61	**64**	18
"Grab me the water bottle."	**30**	47	**83**	27
"Take the screwdriver to the drawer."	**165**	254	**69**	0
"Put the cup in the trash can." (Outdoor)	**46**	63	**67**	18
"Put the bottle in the trash can."	**34**	49	**64**	9