

Bridging Thought and Action: Taming Long-Horizon Instability in Open-Source LLM Agents with a MetaTool-Enhanced ROS Framework

Kazi Abrar Mahmud, Nilotpaul Kundu Dhurubo, Tamal Kirttonia, Sabbir Hossain Ujjal and Mohammad Ariful Haque

Abstract—Large Language Models (LLMs) have enabled more natural human-robot interaction, but open-source models often exhibit unstable long-horizon reasoning and inefficient action execution when deployed in agentic robotic frameworks. This paper presents an enhanced ROS-Agent based architecture that improves task reliability and execution efficiency for agentic robotic systems using open-source LLMs. The proposed system introduces a novel intermediate mechanism, termed the *MetaTool*, which enforces structured planning prior to action execution. Given a natural-language command, the MetaTool induces the LLM to generate a pseudo-code plan of intended tool invocations, which is stored in the ROS-Agent’s scratchpad and persists throughout execution. By explicitly separating planning from execution, the proposed approach reduces execution loops and improves deterministic behavior. The architecture is validated on a custom mobile robotic platform with multimodal perception and motion control capabilities. Experimental results on real-world interactive tasks demonstrate improved task completion and contextual consistency, with up to $\sim 24\%$ gains on complex tasks compared to the baseline framework.

Index Terms—LLMs, Agentic architecture, MetaTool mechanism, Scratchpad memory, Tool-augmented reasoning, Human-robot interaction, ROS integration.

I. INTRODUCTION

RECENT advances in large language models (LLMs) have enabled natural language-based robotic interaction, allowing robots to interpret and execute human instructions through integrated reasoning and tool use. Agentic frameworks that couple LLMs with robotic middleware have emerged as a promising approach for such interaction [1], [2], [3]. In particular, the Robot Operating System Agent (ROSA) framework [4], developed by NASA JPL, provides a structured interface for integrating LLM-based reasoning with ROS-native tools, supporting task decomposition and execution within robotic systems [5], [6].

Kazi Abrar Mahmud is the corresponding author (email: abrar.mahmud790011@gmail.com).

K. A. Mahmud, N. K. Dhurubo, T. Kirttonia, and M. A. Haque are with the Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka 1000, Bangladesh (emails: abrar.mahmud790011@gmail.com; nilbuet60@gmail.com; tamalkirttoniaofficial@gmail.com; arifulhoque@eee.buet.ac.bd).

S. H. Ujjal is with the University of Massachusetts Amherst, 300 Massachusetts Ave, Amherst, MA 01003, United States (email: suj-jal@umass.edu).

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

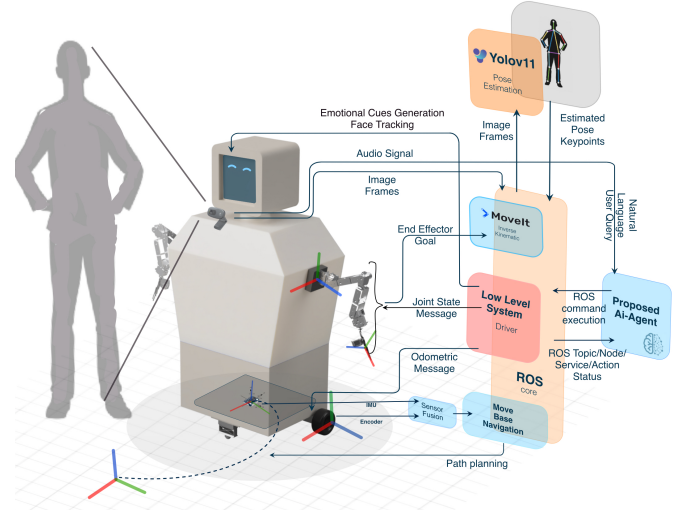


Fig. 1. Overview of the custom-designed robotic platform developed for system evaluation. The platform integrates perception modules, facial emotion display, articulated arm gestures, and mobility components, all managed through a ROS-based control stack.

However, deploying open-source LLMs in agentic robotic pipelines remains challenging. Long-horizon reasoning across multiple tool invocations often leads to instability, including redundant actions, execution loops, and inconsistent task progression [7], [8]. These issues limit execution reliability for complex, multi-step tasks. While proprietary models generally exhibit more stable behavior, their computational cost and limited accessibility motivate architectural solutions that improve the robustness of open-source alternatives. To address these challenges, we propose an enhanced ROS-Agent-based architecture that introduces a novel intermediate mechanism termed the *MetaTool*. The MetaTool operates between the LLM and the ROS execution layer, inducing the model to generate a structured pseudo-code representation of the intended tool sequence prior to execution. This plan is stored in ROS-Agents’s scratchpad memory and persists throughout the observation–reasoning–action loop, explicitly conditioning subsequent tool invocations. The ROS-Agent framework then invokes the selected tools to execute the corresponding robotic actions.

By externalizing task intent into a persistent execution plan, the MetaTool constrains reasoning and action selection, mitigating common failure modes in prompt-driven agentic

TABLE I
FEATURE COMPARISON OF LLM-BASED ROBOTIC PLANNING SYSTEMS.

System	Explicit Plan Buffer	Execution Constraint	ROS-native	Open-LLM Focus	Real Robot
ReAct [10]	×	×	×	×	×
SayPlan [11]	✓	✓	✓	×	✓
RoboGPT [2]	✓	~	×	✓	×
CaP [12]	✓	×	×	×	✓
MetaTool-ROS-Agent (ours)	✓	✓	✓	✓	✓

systems and promoting more predictable execution behavior. We evaluate the proposed architecture through systematic ablation and task-level experiments across multiple models and task complexities, demonstrating consistent improvements in execution reliability, particularly for open-source LLMs. The framework is validated in both ROS–Gazebo simulations [9] and on a custom-built mobile robotic platform, shown in Fig. 1, integrating perception modules, facial emotion display, articulated arm gestures, and mobility within a ROS-based control stack.

Table I compares representative LLM-based robotic planning systems. Unlike prior approaches, the proposed MetaTool ROS-Agent framework combines explicit planning, execution constraints, ROS-native integration, and an open-LLM focus, validated on a real robotic platform.

The remainder of this paper presents related work, methodology, evaluation, and conclusions.

II. LITERATURE REVIEW

Recent advances in large language models (LLMs) have enabled new robot control paradigms that integrate natural language reasoning with embodied action. A representative example is NASA JPL’s ROS-Agent, which bridges ROS-based robots and language interfaces using LLMs to interpret user commands and invoke ROS services through modular tool interfaces [4]. While large closed-source models exhibit strong performance on complex tasks, open-source LLMs often struggle with long-horizon planning unless augmented with structured architectures or explicit memory mechanisms [13].

To improve robustness, many systems augment LLMs with tools or structured outputs. Toolformer trains LLMs to autonomously decide when and how to call external APIs [14], while code-oriented models such as Codex demonstrate that generating executable pseudo-code plans often referred to as “code as policies” can effectively represent multi-step robotic behaviors [12]. These approaches highlight the advantages of explicit, program-like planning for long-horizon tasks.

A large body of recent work explores LLM-driven robotic agents that interleave reasoning and acting or incorporate external tools and memory. ReAct prompts an LLM to alternate between reasoning and action steps within a single loop [10], while Zero-Shot Planners demonstrate that LLMs can decompose high-level instructions into mid-level plans that are subsequently grounded to robot actions [15]. Distinct from purely prompting-based planners, a recent direction fine-tunes open-source LLMs directly on embodied planning data: RoboGPT [2] fine-tunes LLaMA on a large-scale embodied planning dataset to decompose natural-language instructions

into long-horizon subgoal sequences, coupled with a re-planning module to correct infeasible plans. Other systems ground planning in richer representations: SayPlan integrates hierarchical 3D scene graphs with classical planning [11], SPINE validates and refines subtasks in a receding-horizon framework [16], and ELLMER employs retrieval-augmented prompting to generate feedback-aware, adaptive plans [17].

Several works incorporate multimodal feedback or adaptive autonomy. Octopi integrates tactile sensing into a vision-language model to support sequential physical reasoning [18]. CLEAR dynamically adjusts autonomy by prompting an LLM to gate risky actions for human approval [19], while Matcha enables an LLM to iteratively select epistemic actions and update plans using multimodal environmental feedback [20].

Other approaches tightly couple LLMs with task-and-motion planning or specialized control architectures. LLM3 iteratively refines symbolic and continuous plans using motion planner feedback [21], FLTRNN introduces explicit long- and short-term memory to maintain rule-consistent task execution [22], and CoPAL integrates layered reasoning, symbolic planning, and motion generation with replanning for executability and safety [23]. Additional systems include BTGenBot, which generates ROS-compatible behavior trees [24], SayCanPay, which combines LLM proposals with affordance-aware cost evaluation [25], and LABOR, which coordinates bimanual manipulation through explicit LLM planning [26].

Finally, A3VLM explicitly reasons about object articulation and affordances [27], while Manipulate-Anything uses VLMs to autonomously plan, verify, and replan manipulation subgoals [28]. Collectively, these systems demonstrate diverse agentic capabilities, yet long-horizon stability and execution reliability remain open challenges, particularly for open-source LLMs.

III. METHODOLOGY

This section presents the proposed MetaTool-enhanced ROS-Agent architecture and describes its key components, including the agentic reasoning loop, action space, and memory mechanisms.

A. Agentic Architecture, Action Space, and Memory

The proposed system adopts a Reasoning-and-Acting (Re-Act) agentic paradigm grounded in ROS-Agent’s cognitive architecture [4], operating through an iterative observation–reasoning–action loop. As shown in Fig. 2, a user query is forwarded to the LLM together with a prefabricated system prompt that encodes the robot’s embodiment and available

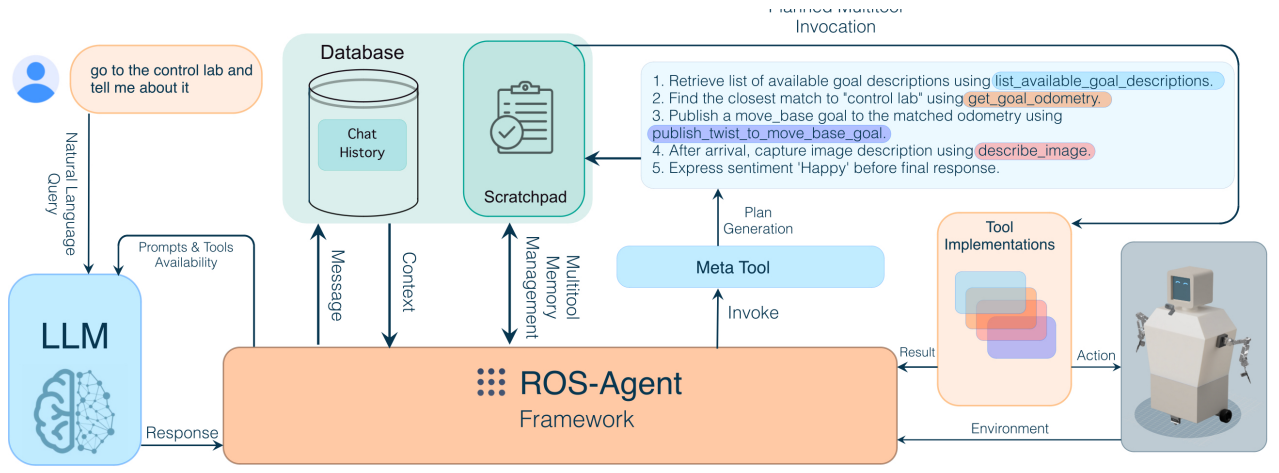


Fig. 2. Functional architecture of the proposed MetaTool-enhanced ROS-Agentic framework.

toolset, constraining the reasoning process to the platform’s operational capabilities. The LLM response is then passed to the ROS-Agent framework, which bridges language-level intent and hardware execution through ROS.

Compared to baseline ROSA, the logic module is extended with a MetaTool-driven planning stage that explicitly separates intent formation from tool execution. Upon the first invocation, the MetaTool generates a structured pseudo-code plan before any task-specific tool is executed. This plan is stored in ROS-Agent’s scratchpad, which is repurposed here as a structured execution buffer that remains accessible throughout the entire interaction episode. Conditioning each tool invocation on this externalized plan, rather than on transient local context, stabilizes the execution trajectory and reduces repetitive or circular tool calls that commonly occur in open-source LLMs during long-horizon, multi-step tasks [29].

The agent’s action space includes ROS-Agent’s native system tools (e.g., *rostopic*, *roscpp*) and several custom task-oriented modules for perception, affective expression, gesture control, and goal-directed navigation. Examples include *describe_image* for vision–language understanding, *face_expression* for social interaction, and *follow_me* for human-following behaviors; additional implemented tools and their functionalities are summarized in Table II.

B. The MetaTool: Structured Reasoning and Execution

The MetaTool represents the central architectural contribution of this work. It functions as an intermediate agentic operator that separates task-level planning from tool-level execution within the reasoning–action loop. Let $q \in \mathcal{Q}$ denote a natural-language user instruction and $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$ the finite set of executable tools available to the agent. The MetaTool first induces the language model to generate an ordered symbolic plan

$$P = \langle a_{i_1}, a_{i_2}, \dots, a_{i_n} \rangle, \quad a_{i_k} \in \mathcal{A},$$

which decomposes q into a sequence of tool calls before any action is executed. Unlike transient chain-of-thought traces,

TABLE II
TASK-ORIENTED TOOLS INTEGRATED INTO THE METATOOL-ENHANCED ROS AGENT

Tool	Functionality
<i>describe_image</i>	Vision–language understanding from live camera input
<i>RAG_query</i>	Knowledge retrieval via vector database querying
<i>face_expression</i>	Affective state publishing for social interaction
<i>follow_me</i>	Human-following behavior using pose keypoints
<i>get_goal_odometry</i>	Goal-conditioned odometry estimation
<i>set_goal</i>	Goal definition and storage for navigation tasks
<i>publish_twist_to_move_base</i>	High-level navigation goal dispatch
<i>stop_ah_v1</i>	Emergency stop and goal cancellation

the plan P is externalized into ROS-Agent’s scratchpad memory and persists throughout the observation–reasoning–action loop.

In a conventional agentic formulation, tool selection is governed by a policy $\pi(a_t | s_t)$, where $s_t \in \mathcal{S}$ denotes the observable state at step t . The MetaTool replaces this with a plan-conditioned execution policy

$$\pi_{\text{meta}}(a_t | s_t, P),$$

which restricts admissible actions to those specified by the plan P . Formally,

$$\pi_{\text{meta}}(a_t | s_t, P) = 0 \quad \forall a_t \notin \mathcal{A}_P(t),$$

where $\mathcal{A}_P(t) = \{a_{i_t}\}$ denotes the plan-specified tool at step t , unless explicit replanning is triggered. Since $|\mathcal{A}_P(t)| \ll |\mathcal{A}|$, this constraint reduces unstructured action branching and prevents recursive loops and instruction drift. As illustrated in Fig. 3, the MetaTool (top-left) is invoked at the start of each

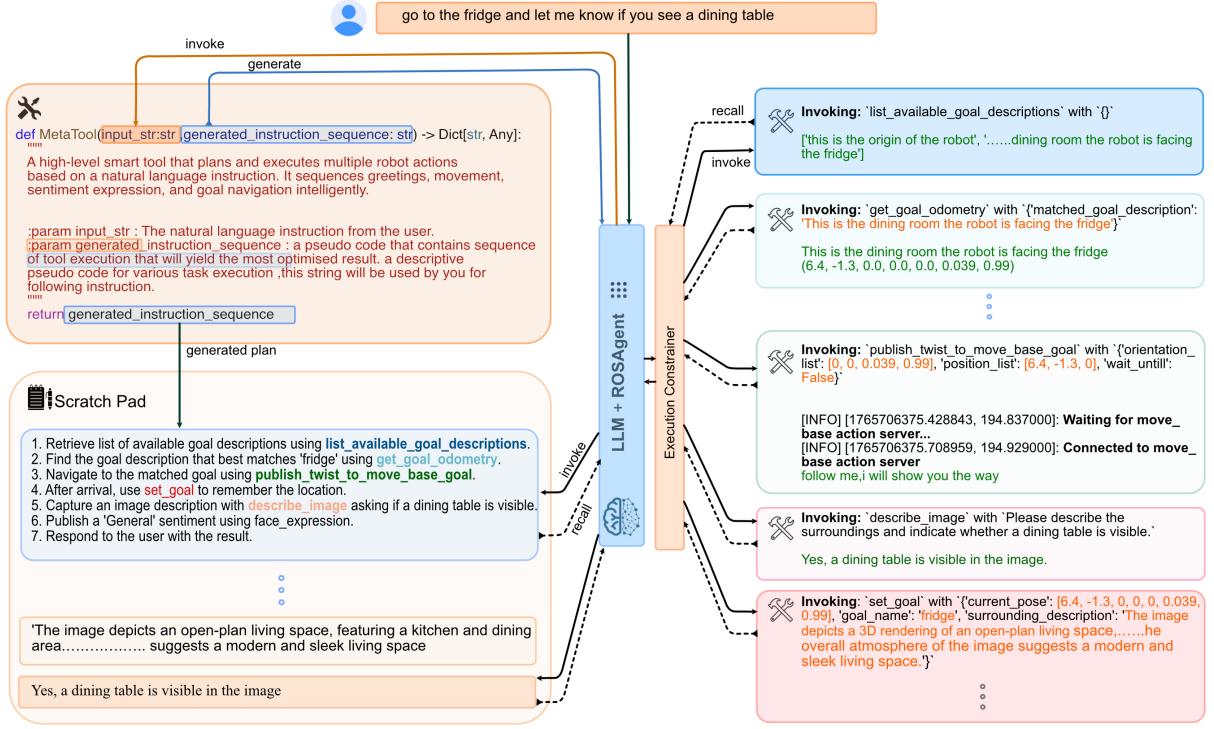


Fig. 3. Operational overview of the proposed MetaTool integrated with the existing ROS agentic architecture, illustrating intermediate task planning and structured tool execution via scratchpad memory.

user query. It generates a structured pseudo-code plan, which is immediately stored in the scratchpad memory. This plan guides all subsequent multi-tool calls, with every tool's output appended to the scratchpad alongside the original plan. By continuously combining the initial plan with the evolving tool outputs (bottom-left of Fig. 3), the scratchpad maintains a coherent execution trace that drives the tool execution chain consistently from start to finish.

The baseline ROS-Agent's scratchpad functions as an unconstrained conversational memory: it records what has occurred, but nothing prevents the agent's subsequent actions from diverging from its prior intent. MetaTool fundamentally redefines this memory as an enforceable execution contract. By formalizing tool selection as a plan-conditioned policy, $\pi_{\text{meta}}(a_t | s_t, P)$, which effectively zeroes out the probability of any action outside the plan's admissible set, $\mathcal{A}_P(t)$ especially for open-source LLMs, as demonstrated in the ablation studies, we transform planning from a passive record into an active constraint on the agent's action space. This reframing mitigates the long-horizon instability commonly observed in agentic systems. Furthermore, this plan-conditioned execution paradigm can, in principle, be adopted by similar frameworks as a drop-in mechanism, positioning MetaTool as an architectural primitive for constraining agentic reasoning under long-horizon uncertainty.

C. Machine Configuration

To evaluate the framework, we developed a cost-effective platform using an Intel-based onboard computer and an NVIDIA GTX 1050 Ti GPU. A custom motor driver fuses IMU and

wheel encoder data for precise odometry. While the LLM operates via the Groq API [30], all perception and control models run locally on off-the-shelf hardware. The system architecture and interconnections are detailed in Fig. 1.

IV. EVALUATION

A. Experimental Setup and Metrics

We evaluated the architecture using two environments: a ROS-Gazebo simulation and a custom physical mobile robot. The system was tested across three task levels (T1–T3) of increasing cognitive complexity, ranging from simple queries to contextually grounded sequential tasks.

- **T1 (Simple Information Query):** Assesses basic knowledge retrieval and NLP parsing. *Example:* "Please brief me about the operational ordinances of this facility. Are there any safety regulations I should follow?"
- **T2 (Complex Contextual Navigation):** Requires semantic reasoning to map high-level functional descriptions to specific locations. *Example:* "Is there a laboratory in this facility that conducts research on power system equipment? If yes, please lead me there."
- **T3 (Contextually Grounded Sequential Task):** Evaluates long-horizon reasoning, human-following, and visual landmark persistence. *Example:* "Please follow me. [Robot follows the user.] Now observe and memorize this area [it includes a distinct red shelf. User introduces a unique landmark.] After some unrelated interactions, the user asks: 'Can you show me the way to the red shelf I showed earlier?'"

TABLE III
COMPONENT-WISE ABLATION RESULTS ACROSS TASKS. TASK IDS ARE DEFINED IN SECTION IV-A.

ID	Configuration	Task	GPT-4o [33]	GPT-OSS-20B [32]	GPT-OSS-120B [32]	Kimi-K2 [34]
C1	ROS-Agent + Adaptive Prompt	T1	15/20 (75%)	11/20 (55%)	11/20 (55%)	15/20 (75%)
		T2	10/20 (50%)	8/20 (40%)	15/20 (75%)	17/20 (80%)
		T3	11/20 (55%)	6/20 (30%)	8/20 (40%)	10/20 (50%)
C2	ROS-Agent + Prompt-Based Planner (Non-Persistent)	T1	17/20 (85%)	12/20 (60%)	15/20 (75%)	15/20 (75%)
		T2	17/20 (85%)	12/20 (60%)	15/20 (75%)	17/20 (80%)
		T3	15/20 (75%)	7/20 (35%)	8/20 (40%)	13/20 (65%)
C3	ROS-Agent + MetaTool (Persistent Scratchpad, No Execution Constraint)	T1	19/20 (95%)	14/20 (70%)	18/20 (95%)	17/20 (85%)
		T2	17/20 (85%)	12/20 (60%)	16/20 (80%)	18/20 (85%)
		T3	15/20 (75%)	7/20 (35%)	10/20 (50%)	13/20 (65%)
C4	ROS-Agent + MetaTool (Persistent Scratchpad + Execution Constraint) (Proposed)	T1	18/20 (90%)	17/20 (85%)	17/20 (85%)	19/20 (95%)
		T2	17/20 (85%)	15/20 (75%)	18/20 (90%)	18/20 (85%)
		T3	15/20 (75%)	11/20 (55%)	14/20 (70%)	14/20 (70%)

While $T1$ focuses on the 'Thought' aspect, tasks $T2$ and $T3$ are designed to evaluate the 'Action' and physical grounding of the MetaTool-enabled agent. We compared the proposed MetaTool-enhanced ROS Agent against the vanilla ROSA baseline [4]. Evaluated models included proprietary LLMs and open-source models (e.g., GPT-OSS-20B, GPT-OSS-120B, Kimi-K2); the latter were selected based on demonstrated tool-invocation competence, evidenced by BFCL performance [31] where available (Kimi-K2: 59.06% overall, Rank 11) and by native function-calling API support and agentic benchmark results [32] for GPT-OSS variants. Collectively, these scores confirm that all three models meet a meaningful threshold of tool-use capability, forming a representative pool of open-source LLMs that spans both efficiency-optimised and reasoning-optimised deployment regimes.

B. Ablation Study

We conducted a component-level ablation study across four configurations ($C1$ – $C4$) to isolate the contributions of persistent planning and execution constraints, as summarized in Table IV. To mitigate the inherent safety risks of autonomous LLM control on physical hardware, the ablation was performed within a Gazebo simulation environment (Fig. 4)¹. The agent was evaluated across diverse environments ,including a large office, a hospital floor, a factory floor with scattered obstacles, and a standard urban home to assess the robustness of its agentic responses. In this setup, $C1$ represents basic prompt engineering; $C2$ incorporates transient, non-persistent planning; $C3$ employs persistent scratchpad planning; and $C4$ (our proposed framework) combines ROS-Agent with MetaTool to enforce execution constraints by conditioning each tool invocation on a persistent task plan.

Rather than comparing distinct architectures, all configurations are evaluated within the same agentic pipeline, differing only in the presence or absence of specific MetaTool components. Quantitative ablation results across tasks and language models are reported in Table III. For each configuration, task success is measured over 20 independent trials per task, and

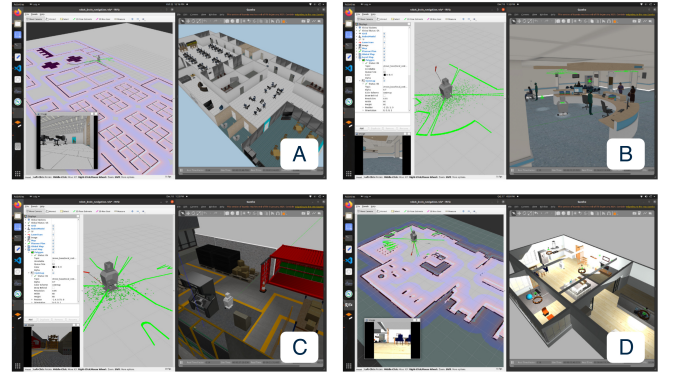


Fig. 4. Gazebo-based simulation and testing across diverse environments to ensure the robustness of agentic responses: (A) large office space; (B) hospital floor environment; (C) factory floor with scattered obstacles; and (D) standard urban home setting.

TABLE IV
METATOOL COMPONENT ABLATION CONFIGURATIONS

ID	Configuration	Description
C1	ROS-Agent + Adaptive Prompt	Natural-language task execution guided solely by adaptive prompting, without explicit planning or persistent state.
C2	ROS-Agent + Prompt-Based Planner (Non-Persistent)	Prompt-based task decomposition generated at runtime; intermediate plans are not stored persistently and may be regenerated.
C3	ROS-Agent + MetaTool (Persistent Scratchpad, No Execution Constraint)	Explicit task plan stored in persistent scratchpad memory; tool selection is not constrained by the stored plan.
C4	ROS-Agent + MetaTool (Persistent Scratchpad + Execution Constraint)	Persistent task plan maintained and used to explicitly condition tool invocation, enforcing adherence throughout execution.

results are reported as empirical success rates. Due to the limited number of trials, these results are interpreted as indicative performance trends rather than statistically significant differences. Key Findings and Analysis:

- 1) **Significance of Persistent Planning ($C1$ vs. $C3$):** Transitioning from basic prompt-engineering ($C1$) to

¹An anonymized codebase for the simulation and proposed execution pipeline is available at: https://anonymous.4open.science/r/uh_bot_v1_sim-BDB1/README.md

persistent scratchpad planning (*C3*) yielded the most consistent performance gains across all LLMs. For GPT-OSS-20B on *T3*, success rate improved from 30% (*C1*) to 35% (*C3*), while GPT-OSS-120B improved from 40% to 50% on the same task demonstrating that persisting the MetaTool-generated pseudo-code in the scratchpad directly mitigates the context-forgetting behaviour common in open-source models during multi-step execution.

- 2) **Impact on Open-Source LLMs (*C2* vs. *C3*):** The benefit of persistent planning over transient prompt-based planning (*C2*) was most pronounced for the open-source models. GPT-OSS-20B improved on *T1* from 60% (*C2*) to 70% (*C3*), and Kimi-K2 improved on *T1* from 75% to 85%. These gains reflect the role of explicit intent externalization in reducing recursive tool invocation and premature action execution failure modes that disproportionately affect smaller-scale models in ROS-native environments.
- 3) **Execution Constraint Stability (*C3* vs. *C4*):** The addition of tool-level execution constraints in the proposed configuration (*C4*) provided measurable stability gains on the most demanding tasks. On *T3*, GPT-OSS-20B improved from 35% (*C3*) to 55% (*C4*), GPT-OSS-120B from 50% to 70%, and Kimi-K2 from 65% to 70%. On *T2*, GPT-OSS-120B reached 90% under *C4* compared to 80% under *C3*. These results confirm that strictly constraining tool invocation to the MetaTool-prescribed plan is critical for reliable convergence in long-horizon, multi-stage navigation tasks.

The ablation results indicate that persistent scratchpad planning and execution constraints are key contributors to improved task success, with the most significant gains observed for open-source models on long-horizon tasks (*T2*, *T3*). Together, these components validate the core design rationale of the proposed MetaTool architecture. Among the open-source models tested, GPT-OSS-120B performed consistently better. Therefore, further evaluation with existing LLM-based robotic planning systems was conducted using GPT-OSS-120B as the LLM backbone.

C. Overall Results and Efficiency Analysis

The MetaTool-enhanced architecture demonstrated superior reasoning stability compared to vanilla ROS-Agent. Fig. 5 compares vanilla ROS-Agent and MetaTool-enabled ROS-Agent across task complexities, highlighting success rates via marker opacity alongside token usage distributions. The high token variance observed in vanilla ROS-Agent indicates that identical queries frequently generate divergent tool sequences, reflecting unstable reasoning and inefficient convergence toward executable plans. In contrast, MetaTool produces compact, low-variance token profiles, demonstrating that it consistently generates structured, executable plans across repeated runs. This stabilizing effect is particularly pronounced in smaller open-source LLMs, where MetaTool effectively limits recursive tool invocations.

To rigorously substantiate these observations, we conducted an empirical evaluation across 40 independent trials on physical robotic hardware using the *T2* task category. Evaluation

was performed using the best performing open-source backbone, GPT-OSS-120B which demonstrated consistent performance among the evaluated open-source LLMs during the ablation study. We recorded average execution time alongside the frequency of repeated and invalid tool invocations.

TABLE V
HARDWARE EVALUATION OF EXECUTION EFFICIENCY AND STABILITY ACROSS 40 TRIALS USING GPT-OSS-120B. ALL METRICS ARE REPORTED AS MEAN $\pm$ STANDARD DEVIATION.

	Repeated Tool Calls (%)	Invalid/Failed Tool Calls (%)	Execution Time (s)
ROS-Agent	19.0 $\pm$ 5.7	21.7 $\pm$ 6.5	89.51 $\pm$ 16.48
MetaTool (ours)	6.1 $\pm$ 2.2	8.1 $\pm$ 2.9	84.23 $\pm$ 9.71

As summarized in Table V, both repeated and invalid tool invocations are reported as the mean $\pm$ standard deviation of their per-trial percentage relative to total tool calls, while execution time denotes task-completion latency in seconds across 40 real-world hardware trials. These empirical results align closely with the token-usage trends illustrated in Fig. 5, confirming that the reduced token variance of MetaTool directly translates to fewer redundant actions, decreased invocation failures, and more consistent execution latency in physical environments.

Finally, to evaluate the proposed framework against existing LLM-based planning paradigms, we conducted a controlled failure mode analysis on the physical robotic platform (shown in Fig. 6), comparing performance against re-implemented ReAct [10] and Code-as-Policies (CaP) [12] baselines. Rather than using the original codebases directly, we re-implemented the core prompting strategy of each within our own agentic pipeline and ROS tool-calling interface. This adaptation was necessary because both paradigms were originally validated on environments incompatible with our platform. ReAct on text-based QA and interactive text environments (HotpotQA, ALFWorld), and CaP on tabletop manipulation in simulation with a distinct action/perception API. This re-implementation preserves each method’s defining mechanism (interleaved reasoning-action with no persistent plan buffer for ReAct and single-shot plan generation without execution constraints for CaP) while grounding them in our specific tool set, task suite (*T1*–*T3*), and physical robot, thereby enabling a direct, controlled comparison against the proposed MetaTool-ROSA architecture under identical conditions.

As detailed in Table VI, the absence of a persistent plan buffer in the ReAct baseline exacerbates recursive reasoning degradation during long-horizon tasks, resulting in a high incidence of incorrect or hallucinated plans (17/40) and navigation failures (13/40), culminating in a low overall success rate of 42.5% (17/40). The CaP-style planner slightly mitigates planning hallucinations (13/40) but remains highly susceptible to tool unavailability and execution timeouts (13/40), limiting its task success to 50.0% (20/40). While the vanilla ROSA pipeline demonstrates a marginal improvement with a 57.5% success rate (23/40 successful trials), it still exhibits significant vulnerability to unconstrained tool invocations, reflected by high rates of hallucinated plans (17/40) and tool timeouts

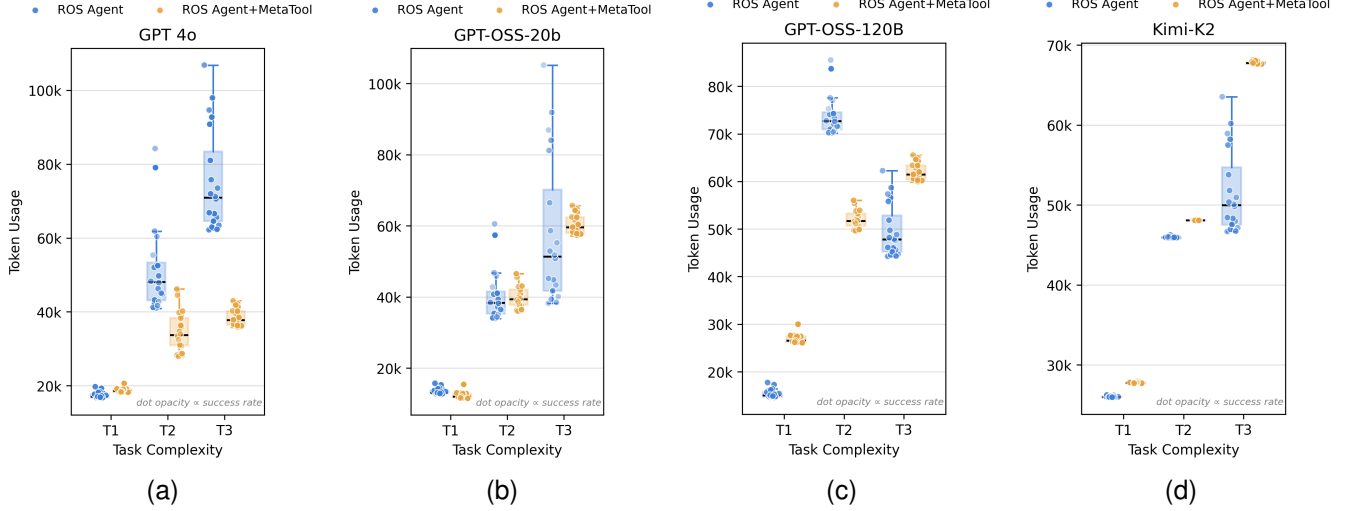


Fig. 5. Comparative Performance Analysis of Baseline ROSA vs. MetaTool-Enabled ROS Agent. The proposed framework demonstrates a higher Success Rate (indicated by the opacity) across all task complexities. The Token Usage Distribution highlights significantly lower variance in the proposed method, indicating that the MetaTool constraint prevents infinite loops and reasoning instability common in the open-source model.

TABLE VI
QUANTITATIVE COMPARISON OF FAILURE MODES AND OVERALL TASK SUCCESS RATES ACROSS EVALUATED PLANNING FRAMEWORKS.

Method	Incorrect / Hallucinated Plan	Navigation Failure	Tool Unavailable / Timeout	Successful Trials
ReAct [10]	17/40	13/40	10/40	17/40
CaP [12]	13/40	10/40	13/40	20/40
ROSA [4]	17/40	10/40	17/40	23/40
MetaTool (ours)	10/40	7/40	3/40	30/40

Note: Failure categories are non-exclusive. Individual failed trials may encounter multiple failure modes, so total category counts may exceed total failed trials.

(17/40). Conversely, our proposed MetaTool framework significantly reduces these failure modes across all categories most notably dropping tool unavailability/timeout to just 3/40 achieving a superior overall success rate of 75.0% (30/40). These empirical findings confirm that unconstrained, prompt-based reasoning mechanisms struggle with the stability demands of real-world robotic deployment. Consequently, the observed failure modes directly underscore the necessity of the persistent scratchpad and structural execution constraints introduced in the MetaTool-ROS-Agent architecture, which successfully translate simulation-based efficiency gains into robust physical operation.

V. CONCLUSION AND FUTURE WORK

This work presented a MetaTool enhanced ROS-Agent architecture for structured agentic task execution in LLM-driven robotic systems. By enforcing an explicit planning stage that generates a pseudo-code task sequence prior to tool invocation, the proposed approach separates high-level reasoning from execution within the ROS-Agent framework. Experimental results demonstrate improved reasoning stability,



Fig. 6. Real-world implementation and experimental evaluation of the proposed architecture on a custom robotic platform: (A) vLM invocation for visual understanding; (B) autonomous navigation within an indoor laboratory environment; (C) visual memorization using agentic vLM calls for future spatial recollection; (D) chauffeuring the user to a requested destination; (E) spontaneous user greeting based on predicted mood and perceived direction of communication; and (F) navigation through a narrow corridor while describing the surroundings to maintain user engagement.

task coordination, and execution reliability for open-source LLMs, particularly in complex multi-step tasks.

Future work will investigate adapting the architecture to smaller open-source language models to enable efficient deployment on resource-constrained robotic platforms. In addition, we plan to extend the robotic platform and tool ecosystem to support broader real-world applications of structured agentic robotic systems.

REFERENCES

- [1] A. Raja and A. Bhethanabotla, "Operatellm: Integrating robot operating system (ros) tools in large language models," in *2024 IEEE 1st International Conference on Communication Engineering and Emerging Technologies (ICoCET)*, 2024, pp. 1–4.
- [2] Y. Chen, W. Cui, Y. Chen, M. Tan, X. Zhang, J. Liu, H. Li, D. Zhao, and H. Wang, "Robogpt: An llm-based long-term decision-making embodied agent for instruction following tasks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 17, no. 5, pp. 1163–1174, 2025.

- [3] L. Sun, D. K. Jha, C. Hori, S. Jain, R. Corcodel, X. Zhu, M. Tomizuka, and D. Romeres, "Interactive planning using large language models for partially observable robotic tasks," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14054–14061.
- [4] R. Royce, M. Kaufmann, J. Bechtel, S. Moon, K. Carpenter, K. Pak, A. Towler, R. Thakker, and S. Khattak, "Enabling novel mission operations and interactions with rosa: The robot operating system agent," in *2025 IEEE Aerospace Conference*. IEEE, 2025, pp. 1–16.
- [5] M. Quigley, "Ros: an open-source robot operating system," in *IEEE International Conference on Robotics and Automation*, 2009. [Online]. Available: <https://api.semanticscholar.org/CorpusID:6324125>
- [6] S. Chitta, I. Sucan, and S. Cousins, "Moveit! [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.
- [7] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:259360665>
- [8] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma, "Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 2081–2088.
- [9] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.
- [10] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafraan, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.
- [11] 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," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 23–72. [Online]. Available: <https://proceedings.mlr.press/v229/rana23a.html>
- [12] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. R. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:252355542>
- [13] H. Xu, Z. Wang, Z. Zhu, L. Pan, X. Chen, S. Fan, L. Chen, and K. Yu, "Alignment for efficient tool calling of large language models," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 17787–17803.
- [14] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 68539–68551. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf
- [15] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 9118–9147. [Online]. Available: <https://proceedings.mlr.press/v162/huang22a.html>
- [16] Z. Ravichandran, V. Murali, M. Tzes, G. J. Pappas, and V. Kumar, "Spine: Online semantic planning for missions with incomplete natural language specifications in unstructured environments," *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13714–13721, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:273162481>
- [17] R. Mon-Williams, G. Li, R. Long, W. Du, and C. G. Lucas, "Embodied large language models enable robots to complete complex tasks in unpredictable environments," *Nature Machine Intelligence*, vol. 7, no. 4, pp. 592–601, 2025. [Online]. Available: <https://doi.org/10.1038/s42256-025-01005-x>
- [18] S. Yu, L. Kelvin, and H. Soh, "Demonstrating the Octopi-1.5 Visual-Tactile-Language Model," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025.
- [19] J. P. Macdonald, R. Mallick, A. B. Wollaber, J. D. Peña, N. McNeese, and H. C. Siu, "Language, camera, autonomy! prompt-engineered robot control for rapidly evolving deployment," in *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 717–721. [Online]. Available: <https://doi.org/10.1145/3610978.3640671>
- [20] X. Zhao, M. Li, C. Weber, M. B. Hafez, and S. Wermter, "Chat with the environment: Interactive multimodal perception using large language models," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3590–3596, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:257532768>
- [21] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, "Llm3: Large language model-based task and motion planning with motion failure reasoning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 12086–12092.
- [22] J. Zhang, L. Tang, Y. Song, Q. Meng, H. Qian, J. Shao, W. Song, S. Zhu, and J. J. Gu, "Fltrnn: Faithful long-horizon task planning for robotics with large language models," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6680–6686, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:271798999>
- [23] F. Joubin, A. Ceravola, P. Smirnov, F. Ocker, J. Deigoeller, A. Belardinelli, C. Wang, S. Hasler, D. Tanneberg, and M. Gienger, "Copal: Corrective planning of robot actions with large language models," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8664–8670, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:263835008>
- [24] R. A. Izzo, G. Bardaro, and M. Matteucci, "Btgenbot: Behavior tree generation for robotic tasks with lightweight llms," *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9684–9690, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:268531996>
- [25] R. Hazra, P. Z. D. Martires, and L. D. Raedt, "Saycanpay: Heuristic planning with large language models using learnable domain knowledge," *ArXiv*, vol. abs/2308.12682, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:261100610>
- [26] K. Chu, X. Zhao, C. Weber, M. Li, W. Lu, and S. Wermter, "Large language models for orchestrating bimanual robots," *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, pp. 328–334, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:268856981>
- [27] S. Huang, H. Chang, Y. Liu, Y. Zhu, H. Dong, P. Gao, A. Boularias, and H. Li, "A3vlm: Actionable articulation-aware vision language model," *ArXiv*, vol. abs/2406.07549, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:270379695>
- [28] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna, "Manipulate-anything: Automating real-world robots using vision-language models," in *Proceedings of The 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, P. Agrawal, O. Kroemer, and W. Burgard, Eds., vol. 270. PMLR, 06–09 Nov 2025, pp. 5326–5350. [Online]. Available: <https://proceedings.mlr.press/v270/duan25a.html>
- [29] K. Zhu, Z. Liu, B. Li, M. Tian, Y. Yang, J. Zhang, P. Han, Q. Xie, F. Cui, W. Zhang *et al.*, "Where llm agents fail and how they can learn from failures," *arXiv preprint arXiv:2509.25370*, 2025.
- [30] Groq Team, "Groqcloud api documentation," 2024, accessed: 2026-01-09. [Online]. Available: <https://console.groq.com/docs/>
- [31] S. G. Patil, H. Mao, C. Cheng-Jie Ji, F. Yan, V. Suresh, I. Stoica, and J. E. Gonzalez, "The berkeley function calling leaderboard (bfc1): From tool use to agentic evaluation of large language models," in *Forty-second International Conference on Machine Learning*, 2025.
- [32] S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, E. Arbus, R. K. Arora, Y. Bai, B. Baker, H. Bao *et al.*, "gpt-oss-120b & gpt-oss-20b model card," *arXiv preprint arXiv:2508.10925*, 2025.
- [33] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford *et al.*, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.
- [34] K. Team, Y. Bai, Y. Bao, G. Chen, J. Chen, N. Chen, R. Chen, Y. Chen, Y. Chen, Y. Chen *et al.*, "Kimi k2: Open agentic intelligence," *arXiv preprint arXiv:2507.20534*, 2025.