A Glimpse into Long-term Physical Coexistence with Intelligent Robots

Astribot Team

research@astribot.com

Project Page: www.astribot.com/research/Philia

Author List in Contributions

arXiv:2607.11377v1 [cs.RO] 13 Jul 2026

Abstract

Long-term physical coexistence with intelligent robots requires more than capable robot policies. A persistent robotic assistant must support diverse user-facing interfaces, maintain long-horizon memory of people and preferences, coordinate across robot embodiments, and translate open-ended human intent into safe physical execution. We introduce PHILIA, a multi-robot agent built around a robot gateway abstraction. PHILIA retains the rich interaction and tool ecosystem of OpenClaw, while exposing robot-local runtimes, onboard perception, navigation, speaker, and robot policies through a unified capability interface. This design decouples low-frequency, high-semantic agent reasoning from high-frequency, low-level robot execution, enabling plug-and-play integration of user interfaces, robot embodiments, and policy backends. As a result, the user experience becomes compositional: advances in user interfaces, robot embodiments, robot policies, navigation stack, or interaction algorithms can improve the overall experience without requiring a redesign of the full system. We validate the agentic architecture on Astribot S1 robots, while designing the robot gateway contract to support future heterogeneous robot platforms through a shared capability interface for observation, task execution, navigation, speech playback, status monitoring, and task cancellation. We present representative use cases in which agent memory and scene understanding are grounded in physical robot actions. These use cases span interactive, open-ended household scenarios, ranging from simple organization to more challenging long-horizon and dexterous service tasks, such as packing the backpack and lifting the garbage bag. We highlight the human–robot interaction flow, where contextual understanding of user intent and preferences, together with human-in-the-loop confirmation or adjustment during execution, is essential for the robot to provide effective assistance. We hope this report serves as an invitation to imagine, explore, and shape the future of human–robot coexistence and co-creation together.



1. Introduction

Humans have long envisioned robots that can coexist with us in everyday environments: reliable enough to share our spaces, helpful enough to assist with open-ended requests, and socially intelligible enough to participate in daily interactions. Recent advances in both generalist and specialized robot policies have substantially expanded the capabilities of robotic systems (Black et al., 2024; Brohan et al., 2022; Kim et al., 2024; Tang et al., 2025; Team et al., 2024; Zitkovich et al., 2023). However, policy execution alone is insufficient to realize an intelligent robotic companion. Such a system must be capable of interacting through diverse interfaces, maintaining long-term memories of people and past interactions, coordinating across multiple physical embodiments, and safely translating open-ended human intentions into grounded physical actions (Ahn et al., 2022, 2024; Driess et al., 2023; Wu et al., 2023).

Most deployed robotic systems are designed around isolated command execution, where users invoke individual skills through robot-specific interfaces. While effective for standalone tasks, this paradigm does not scale to long-term human–robot coexistence or multi-robot environments that require seamless integration of heterogeneous robots, policy servers, navigation systems, user interfaces, and communication channels (Hawes et al., 2017; Liu et al., 2023; Sarkar et al., 2025; Zhang et al., 2025). Moreover, robotic capabilities span fundamentally different temporal and semantic scales: language interaction, planning, and memory operate at a low frequency and high level of abstraction, whereas perception, navigation, manipulation, and safety-critical control demand high-frequency execution grounded in the physical world. We therefore advocate decoupling the robotic assistant from any specific robot embodiment, treating it as a persistent control plane responsible for user intent, memory, semantic context, tool usage, and robot orchestration, while robot-local runtimes handle perception, navigation, policy execution, low-level control, and safety-critical operations. This separation enables a unified assistant identity to coordinate multiple robot platforms while maintaining the reliability, scalability, and auditability of real-time execution (Colledanchise and Ögren, 2018; Macenski et al., 2022;

Quigley et al., 2009).

In this work, we introduce PHILIA, an agentic multi-robot architecture designed for long-term physical coexistence with intelligent robots. At its core, PHILIA adopts a robot gateway abstraction that exposes a compact set of high-level capabilities while encapsulating platform-specific middleware, policies, and control logic. Above these gateways, an OpenClaw-based control plane provides unified interaction, memory, planning, authorization, and actor-aware task routing (Steinberger and Contributors, 2026). Unlike prior embodied-AI systems that primarily focus on planners, foundation models, or single-robot demonstrations (Huang et al., 2023; Kim et al., 2024; Liang et al., 2023; Team et al., 2024), PHILIA investigates the runtime architecture required to deploy and coordinate these components in real-world settings. Specifically, it enables robot capabilities to be exposed, invoked, monitored, interrupted, recovered, and orchestrated across multiple robot actors through a single assistant identity. By decoupling agent intelligence from robot-local execution, PHILIA establishes a stable, auditable, and scalable control plane that makes robotic capabilities operational, composable, and transferable across heterogeneous embodiments.

This architecture enables agent intelligence to enhance robot behavior without modifying robot-local policies. Rather than encoding task semantics directly into policies, the agent provides semantic grounding, memory, and task composition, allowing the same execution backend to operate in richer contexts. For example, an agent can infer scene-specific organizing rules from observations before invoking a generic pick-and-place policy, or leverage long-term user preferences to select appropriate items when coordinating physical tasks. More broadly, the architecture supports compositional improvement: advances in user interfaces, robot embodiments, policies, perception, navigation, memory, or dialogue systems can independently enhance the overall assistant experience. Because these components interact through stable capability and gateway abstractions, improvements in any subsystem can be integrated without redesigning the entire stack, enabling scalable and continuously evolving robot assistants.

The main contributions of this work are:

- We propose an agent–runtime decoupled architecture that separates low-frequency semantic reasoning from high-frequency robot execution, enabling scalable, long-term operation across heterogeneous robot embodiments.
- We introduce **robot gateways**, a capability-centric abstraction that decouples agent intelligence from robot-specific implementations, allowing semantic reasoning, memory, and planning capabilities to generalize across diverse robots without modifying robot-local execution stacks.
- We formulate robotic assistants as a compositional system in which advances in user interfaces, memory systems, robot embodiments, navigation modules, and policy backends can be integrated independently behind stable capability boundaries, yielding cumulative improvements in overall user experience.
- We validate the architecture through real-world deployments on the Astribot S1 platform and demonstrate that robot-local policies can be continuously improved while preserving the same capability contracts and system interfaces.

2. PHILIA

2.1. Preliminaries

PHILIA is built around three abstractions: a persistent user-facing assistant, robot actors, and robot-local capabilities. The assistant maintains the conversational identity, long-term memory, user context, tool ecosystem, and high-level task planning. Robot actors are physical embodiments that execute the assistant's intentions through a shared capability interface, enabling heterogeneous platforms with different sensing, manipulation, navigation, or interaction capabilities to be integrated without modifying the assistant. Robot-local capabilities expose high-level operations, such as scene observation, policy execution, semantic navigation, speech output, and task interruption, while encapsulating robot-specific implementations behind a stable gateway interface.

This architecture decouples semantic reasoning from robot execution. The assistant is responsible for intent understanding, memory retrieval, tool use, and task composition, whereas robot runtimes handle real-time perception, control, policy inference, navigation, and safety-critical execution. Communication between the two is mediated through a compact gateway interface, allowing each layer to evolve independently.

Human–robot interaction follows an interleaved planning–execution paradigm. Long-horizon and compositional user requests are decomposed into plans spanning both robot and non-robot actions, with execution interleaved with perception such that intermediate observations continually inform subsequent decisions. In contrast, simple, bounded requests—including status queries, scene observations, and human interventions (e.g., stop commands or minor action adjustments)—are executed directly without explicit planning. Both execution modes share the same actor resolution, authorization, capability abstraction, and gateway interface, providing a unified interaction model across heterogeneous robot embodiments.

2.2. Agentic System Architecture

System Overview. As illustrated in Fig. 1, PHILIA adopts a three-layer architecture consisting of user interfaces, an agent control plane, and robot-local gateways. User interfaces provide interaction channels, including messaging, voice, and web applications. The agent control plane maintains persistent assistant sessions, resolves target robot actors and their capabilities, enforces authorization and safety policies, and dispatches requests to the appropriate robot gateways. Robot-local gateways expose high-level capabilities while encapsulating platform-specific execution details, including middleware, sensing, navigation, bridge control, execution state, and robot policies. By cleanly separating semantic reasoning from robot-local execution, the architecture enables heterogeneous robot platforms to be integrated behind a stable capability interface, allowing new robot embodiments and policy implementations to be incorporated without modifying either the agent or the underlying robot software.

Compositional User Experience. The stable boundaries between user interfaces, the agent control plane, robot actors, and robot-local capabilities decouple system components, allowing each to evolve independently. Improvements to user interfaces, robot embodiments, reasoning and memory, perception, navigation, or robot-local policies can be incorporated without changing the surrounding system interfaces. As a result, advances at any layer directly translate into



Figure 1 | **System Architecture Illustration.** PHILIA unifies user-facing interfaces, an agent control plane, and heterogeneous robot-local gateways through a shared capability abstraction.

improved user experience, enabling cumulative system-wide progress through compositional integration rather than end-to-end redesign.

Low-frequency Reasoning and User-facing Interaction PHILIA extends OpenClaw with physical embodiment while preserving its existing assistant infrastructure, including multi-modal user interfaces, tool invocation, session management, UI integration, media handling, and extensible runtime hooks. Robot interaction is treated as a first-class assistant capability rather than a separate execution mode. Within a single assistant session, the system seamlessly interleaves conversational interaction, memory retrieval, tool use, robot perception, and physical task execution through robot actors.

Agent Control Plane. The agent control plane operates entirely at the semantic level. It resolves target robot actors from user-facing names and aliases, discovers their advertised capabilities, validates capability invocations against authorization and readiness constraints, constructs execution plans for compound requests, and tracks progress across multiple actors. Free-form user instructions are interpreted through a capability-grounded interface that provides the agent with the runtime capability manifest and produces structured dispatch decisions. To balance responsiveness with robustness, request routing follows a five-stage cascade: regex fast-pass, local classifier gate, deterministic router, capability-grounded model router, and agent execution. Instead of generating low-level motor commands, the control plane emits structured capability requests that are executed by robot-local gateways. This decoupling allows semantic reasoning, planning, and long-term memory to be shared across heterogeneous robots while preserving a compact, auditable, and platform-independent execution interface.

Plans and Workflows. Bounded interactions, such as scene observation, status queries, or explicit stop commands, are dispatched directly as individual capability invocations. Compound goals are first decomposed into workflows spanning robot and non-robot actions, with each step corresponding to a capability invocation. The agent control plane schedules execution, incorporates intermediate observations to guide subsequent decisions, and handles retries, recovery, or termination based on action outcomes. Task composition therefore remains an assistant-side responsibility, while each capability is executed independently by the robot-local runtime. This separation enables flexible high-level reasoning over *what* to do while delegating *how* to execute each step to robot-local policies and control systems.

Robot Gateway Abstraction. A robot gateway defines the robot-local interface exposed to the assistant. Each gateway publishes a runtime manifest describing the capabilities supported by its robot, together with the corresponding input and output schemas. Because manifests are discovered dynamically, newly deployed policies or capabilities become immediately available to the agent without modifying the control plane. Behind this interface, robots may employ different middleware, SDKs, navigation systems, policy implementations, or bridge processes. The control plane remains agnostic to these implementation details, interacting only with the capabilities advertised by each gateway. The gateway interface is intentionally positioned between a low-level driver API and a full robot middleware. It does not impose a common sensor suite, action space, controller, SDK, or navigation stack. Instead, each robot advertises the capabilities it supports while encapsulating platform-specific implementation details behind the gateway. This abstraction enables a single assistant to coordinate heterogeneous robot platforms while preserving robot-specific optimization within each local runtime.

Unified Multi-Robot Interface. PHILIA represents robot embodiments as actor-scoped instances of a shared assistant identity. An actor registry maps user-facing names to robot actors, a backend registry associates actors with robot-local gateways, and a capability catalog advertises the operations each actor supports, decoupling user interfaces, assistant logic, and robot execution. The same abstraction extends to spatial context: the control plane reasons over shared semantic place names, while each robot maintains its own gateway, navigation stack, and local map. This design enables heterogeneous robots to operate within a common semantic environment while preserving platform-specific execution, localization, and navigation. Although our current implementation targets Astribot S1 robots, additional platforms can be incorporated simply by registering new actor–gateway pairs and their capability manifests.

Capability Exposure. Manipulation policies are exposed as robot-local capabilities. The agent performs semantic reasoning, including task interpretation, policy selection, prompt generation, and coordination with perception and navigation, while the robot-local runtime handles image processing, action generation, and high-frequency control. This separation enables open-ended agent reasoning to enhance policy usability without requiring manipulation policies to incorporate interaction, memory, or task management.

Semantic State and Long-Horizon Context. State is partitioned across architectural layers according to its semantics and update frequency. The agent control plane maintains persistent semantic state, including user preferences, long-horizon plans, semantic place names, and task history, while robot-local runtimes retain high-frequency execution state, such as sensor

streams, maps, coordinate frames, and policy buffers. Robot gateways exchange only lightweight semantic summaries, including capability readiness, place manifests, and task outcomes. This layered state abstraction enables the assistant to reason over persistent semantic concepts while delegating real-time perception, localization, and execution to robot-local runtimes.

Safety Envelope. Physical actions are mediated through a small set of explicit execution gates rather than relying solely on dialogue context. Before dispatch, the control plane enforces authorization, confirmation, and readiness checks based on the gateway's published state (e.g., connectivity, localization, posture, and map availability). During execution, actor-scoped task arbitration guarantees that at most one motion-changing capability is active per robot, with preemption or rejection handled according to explicit policies. Stop and cancel operations are likewise scoped to individual actors, ensuring that interruptions affect only the intended robot. These safeguards preserve the flexibility of natural-language interaction while providing a narrow, auditable, and safety-conscious path for physical execution.

2.3. Policies as Capabilities.

Recent advances in robot learning have significantly expanded the capability boundaries of robotic systems. We briefly summarize the learning strategies adopted to acquire and continuously improve these capabilities, training from a mixture of expert demonstrations, autonomous rollouts, and corrective interventions. During deployment, we employ training-time real-time chunking together with runtime trajectory filtering to mitigate temporal inconsistencies between training and execution. Throughout this process, action generation, action-chunk execution, and real-time control remain within the robot-local runtime, while the assistant continues to invoke the same stable capability interface as the underlying policies evolve.

Foundation Models as Zero-Shot Executors. Recent advances in robot learning have rapidly improved generalist robot policies and large-scale cross-embodiment datasets, substantially expanding the range of tasks that robots can perform once an appropriate objective is specified. In particular, recent foundation models have begun to demonstrate promising zero-shot capabilities on simple manipulation tasks, such as picking and placing common objects, opening and closing drawers, and manipulating common household objects (Tang et al., 2025). PHILIA is designed to seamlessly benefit from these advances by exposing robot policies through a stable capability interface, allowing improved policies to be composed with memory, user interfaces, navigation, and safety mechanisms while preserving the same assistant-facing interface for long-term autonomous assistance.

Task-Specific Supervised Fine-Tuning (SFT). For tasks requiring greater dexterity or long-horizon coordination, we collect expert demonstrations to fine-tune a generalist foundation policy, yielding reliable performance within the distribution represented by the demonstrations. We refer readers to (Larchenko et al., 2025) for common supervised fine-tuning (SFT) techniques. The resulting policy provides a strong initialization for deployment, serving as both an immediately deployable controller and a stable, comparatively safe behavioral prior for subsequent real-robot data collection and iterative policy improvement.

Closing the Deployment Loop. For high-complexity tasks, we further improve deployed policies through online or offline reinforcement learning (RL). In the offline setting, we adopt advantage-conditioned post-training to compensate for the limited coverage of failure and recovery states in demonstration-only supervised fine-tuning (SFT). Expert demonstrations rarely contain corrective behaviors for grasp failures, abnormal object configurations, motion stagnation, or repetitive local behaviors, although such states frequently arise during closed-loop execution and can lead to compounding out-of-distribution errors once the policy deviates from the demonstration manifold. Following prior intervention-based real-robot post-training paradigms (Intelligence et al., 2025; Kelly et al., 2019; Yu et al., 2026), we collect autonomous rollouts from the deployed SFT policy and augment them with corrective human interventions when necessary, thereby enriching the training distribution around failure states. Rather than treating these trajectories as additional SFT data, we perform advantage-conditioned post-training by training an advantage estimator on a frame-level progress signal derived from each trajectory's normalized temporal index. The estimator predicts the relative task progress between the current and future states, and its continuous outputs are converted into task-specific binary optimality labels through percentile thresholding. The policy is then conditioned jointly on the task instruction and the optimality label, enabling it to distinguish high-utility behaviors from suboptimal alternatives under the same task semantics. This procedure performs offline policy extraction rather than online policy-gradient optimization: the SFT policy provides the underlying action manifold, rollout and intervention data expand recovery-state coverage, and advantage conditioning biases the learned policy toward behaviors with higher task utility while preserving the stability of supervised learning.

One practical challenge in this SFT-plus-offline post-training pipeline is the mismatch between data semantics and training granularity. In our implementation, naive frame-level advantage labels are often inconsistent with 32-step action chunks: adjacent frames supervise highly overlapping future action sequences but may receive alternating positive and negative labels, introducing contradictory supervision for nearly identical action targets. To reduce this source of variance, we perform chunk-aware binarization by aggregating frame-level advantage scores within each action chunk before thresholding and, when necessary, applying temporal smoothing to suppress label oscillation. Human interventions provide valuable coverage of failure and recovery states, but are not assumed to be high-advantage samples by default. Before rollout data is incorporated into training, we analyze its statistics and visualize representative trajectories to verify that it captures reproducible recovery behaviors rather than merely increasing the diversity of noisy or suboptimal data.

Smooth Execution. Contemporary robot policies generate actions in fixed-length chunks, while asynchronous inference introduces latency such that a portion of the previous chunk may already have been executed before the next prediction becomes available. If this committed action prefix is not modeled during training, consecutive chunks can become temporally misaligned, particularly in states where similar observations admit multiple plausible action continuations. Following the training-time real-time chunking formulation of (Black et al., 2025b), which extends earlier work on asynchronous execution and action-chunk handling (Black et al., 2025a), we sample a random inference delay during training, treat the corresponding executed prefix as conditioning, and optimize the loss only over the remaining postfix actions. The action prefix serves as a short-horizon execution context that is maintained and injected by the robot-local runtime rather than the assistant. During deployment, we further apply real-time trajectory filtering in the spirit of (Gao et al., 2025) to improve trajectory continuity across chunk boundaries. Unlike the training objective, this runtime procedure filters and smooths consecutive action

chunks without modifying the learned policy. Together, training-time real-time chunking and runtime trajectory filtering provide a temporal alignment mechanism between post-training and real-world execution: the former conditions policy prediction on committed actions, while the latter mitigates discontinuities introduced by asynchronous chunk switching.

2.4. Memory

2.4.1. Agentic Memory

In the current system, long-term semantic memory is inherited from OpenClaw's Markdown-based memory framework. Durable user facts, preferences, and persistent instructions are stored in curated long-term memory files, while day-to-day interaction context is recorded as dated Markdown notes. These files serve as the assistant-level source of truth for persistent semantic knowledge. The use of Markdown is deliberate: it keeps memory human-readable, editable, portable, and fully inspectable, rather than embedding persistent state within opaque model parameters. To maintain efficiency, the assistant does not load the entire memory corpus into each interaction. Instead, OpenClaw provides retrieval tools that first identify relevant memory entries through semantic search and then selectively read only the required snippets. This retrieval-on-demand design preserves a compact working context while enabling previously acquired preferences, decisions, semantic locations, task history, and other long-term knowledge to be incorporated into reasoning whenever they become relevant.

PHILIA builds upon this mechanism as a high-level bridge between long-term memory and physical action. Retrieved memories are not interpreted as direct robot commands or as the sole source of physical truth; instead, they provide semantic context for the agent control plane. For example, if a user has previously recorded a preference for sugar-free beverages or a low-calorie diet, a later request such as preparing afternoon tea can trigger retrieval of the relevant memory. The agent incorporates this information into task planning, queries the robot for the current scene, and formulates appropriate gateway-mediated perception and manipulation requests. This establishes a controlled path from memory to physical action: long-term semantic memory informs the interpretation of user intent, while the agent integrates the retrieved context with current observations, actor availability, capability manifests, and safety constraints before dispatching a structured request through the robot gateway. Robot-local runtimes remain responsible for perception, navigation, policy execution, and real-time safety checks. Consequently, memory enhances personalization and long-horizon task continuity without bypassing the gateway abstraction or allowing stored text to directly influence low-level robot control.

2.4.2. Policy Memory

Most existing policies model manipulation tasks as standard Markov Decision Processes (MDPs), an assumption that carries inherent limitations for long-horizon tasks reliant on temporal context. During training, the absence of sufficient sequential history as conditioning signals deprives the optimization process of structured temporal constraints, which readily leads to action space collapse and unstable policy convergence. At deployment, reliance on single-frame observations undermines the model's ability to accurately infer the current task phase, frequently resulting in suboptimal or erroneous decision-making.

To address this limitation, PHILIA incorporates two complementary history injection mecha-

nisms. The first approach employs fixed-window historical action tokens, where a unified action encoder compresses and encodes the proprioceptive action sequences from the preceding 128 timesteps into compact latent representations. The second approach projects historical action trajectories onto the robot's primary head view, embedding temporal history directly into visual observation encodings. This minimally invasive design achieves bidirectional alignment and unification of historical information across both vision tokens and language tokens.

We elaborate on the visual trajectory injection scheme as follows. We take state sequences spanning 128 timesteps prior to the current timestep t as historical input. Using the intrinsic and extrinsic parameters of the primary-view camera, we compute the historical trajectories of the left and right gripper end-effectors in the camera frame and overlay these trajectories onto the visual observation. A gradual color gradient along the rendered trajectories encodes temporal order, constructing strong visual history cues within the observation tokens.

2.5. Navigation

Navigation is exposed as a standard robot-local capability through the unified gateway interface, consistent with manipulation and observation. The agent reasons only over semantic places and high-level task goals, while mapping, localization, motion planning and safety validation remain fully encapsulated within the robot-local runtime. In a typical composed workflow, the robot first navigates to a target semantic place, performs local alignment to reach the policy execution pose, and then executes manipulation policies. This design keeps spatial grounding close to the robot while allowing the agent to compose navigation with interaction and memory.

Site Mapping and Semantic Place Annotation. A deployment site is mapped via an operator-guided session: the robot traverses the environment while its on-board SLAM stack builds a persistent map. The operator marks a small set of semantic places (e.g., home dock, workstations) and a dedicated calibration anchor, all recorded in the persistent map frame to preserve geometric consistency across runtime restarts. The resulting place manifest is published to the control plane as named semantic references, without exposing raw map data or low-level pose state to the agent.

Per-Session Re-localization Calibration. As the odometry frame is re-initialized on every navigation runtime startup and misaligned with the saved map frame, a lightweight calibration step bridges this gap. The operator drives the robot to the physical calibration anchor and triggers re-localization; the runtime captures the current odometry pose, retrieves the saved anchor pose, and computes an in-memory rigid transform between the two frames. All subsequent navigation requests apply this transform to saved place poses before planning. The agent only consumes a calibration status flag, while the transform itself remains a robot-local artifact.

Multi-Robot Composability. Since semantic places are anchored in a shared map frame and re-localization runs independently per robot actor, a single mapped environment can serve multiple actors. The persistent map and place manifest are scoped to the deployment site and shared across all actors, while each robot maintains its own navigation runtime and calibration transform. Adding a new robot to an already mapped site requires only re-localization at the anchor, with no re-mapping or per-robot place re-authoring needed. This separation of

map-level semantics (control plane) and live localization (robot runtime) forms the foundation of composable multi-robot navigation.

Local Control and Safety Boundary. Bounded local motions, including small chassis adjustments and recovery maneuvers, are also exposed as actor-scoped gateway capabilities under the same safety envelope as other physical actions. The system does not grant the language model direct access to continuous motion control, maintaining responsiveness in daily interaction while upholding safety guardrails.

3. Experiments

Our evaluation focuses on the control-plane dispatch boundary that bridges free-form user language and robot capabilities. Specifically, we examine whether requests are routed reliably, whether the dispatch mechanism generalizes across different language models, and how inference cost can be controlled. We regard this dispatch boundary as the central empirical component of the control-plane architecture (§2), since every robot action—regardless of the underlying backend or manipulation policy—is ultimately mediated through it.

3.1. Capability-Grounded Dispatch

Setup. We evaluate routing performance on a balanced test set of N=483 utterances covering four intent classes: `robot_task` (physical actions), `robot_query` (current state observations), `robot_control` (preemptive control such as stop, pause, resume, or safety override), and `non_robot` (general dialogue and configuration). The dataset mixes formal and colloquial phrasing, Chinese and English, and explicit and implicit robot references. We compare two router configurations across seven language models of varying parameter scales: (1) a lightweight binary gate that classifies whether an utterance is robot-related; (2) a full capability-grounded router that outputs the matched capability ID, target actor, and missing context.

Core findings. As summarized in Tab. 1, the evaluation yields three key conclusions:

1. **Model-agnostic routing reliability.** The framework delivers consistent performance across all seven tested models, from small on-device models to frontier large systems. The lightweight binary gate achieves F1 $\geq$ 0.96 and recall $\geq$ 0.98 on every model. The local Qwen3.5-4B completes lightweight inference in only 414 ms, making it suitable for fast pre-filtering.
2. **High-precision full dispatch on frontier models.** For the capability-grounded configuration, frontier models reach 89–94% top-1 skill matching accuracy on robot requests, and 83–89% accuracy on explicit actor resolution. Claude-series models achieve a 100% rejection rate on non-robot utterances, eliminating false-positive physical actions.
3. **Zero-overhead skill extensibility.** The grounded router draws candidates from a runtime manifest catalog, so new skills become routable immediately after manifest publication, with no changes to the routing prompt or model required.

Model	Config	Binary F1	Cap top-1	Non-R reject	Actor	p50 (ms)
gpt-5.4 (OpenAI, 2026a)	lightweight	0.994	—	—	—	3 737
	grounded	0.984	0.893	0.991	0.862	9 151
gpt-5.5 (OpenAI, 2026c)	lightweight	0.989	—	—	—	2 933
	grounded	0.991	0.919	0.991	0.892	4 710
gemini-3.5-flash (Google, 2026)	lightweight	0.989	—	—	—	6 930
	grounded	0.991	**0.939**	0.983	0.877	10 690
claude-opus-4.6 (Anthropic, 2026)	lightweight	0.988	—	—	—	2 673
	grounded	0.981	0.919	**1.000**	0.831	3 609
claude-opus-4.8 (Anthropic, 2026)	lightweight	0.988	—	—	—	2 875
	grounded	0.979	0.904	**1.000**	0.831	3 789
gpt-5.4-mini (OpenAI, 2026b)	lightweight	0.985	—	—	—	2 532
	grounded	0.989	0.803	0.974	0.508	4 459
Qwen3.5-4B (Qwen Team, 2026)	lightweight	0.965	—	—	—	**414**
	grounded	0.962	0.883	0.809	0.277	10 585

Table 1 | **Dispatch quality across 7 models and 2 routing configurations.** Lightweight configuration: binary robot/non-robot F1 for gate reliability. Capability-grounded configuration: top-1 skill matching (198 robot cases; 115 non-robot rejection cases) and actor resolution accuracy (65 explicit mention cases). Latency is per-call p50.

3.2. Latency Decomposition and the Cascaded Routing

Dispatch quality results motivate the cascaded design: the full capability-grounded router delivers higher precision but incurs 1.4× latency on the deployed claude-opus-4.6, and up to 25× on the smallest model, as shown in Tab. 1. Running the grounded router directly on a small local model yields only modest full-dispatch performance at 10.6 s latency, as it must process the complete manifest catalog and output structured results.

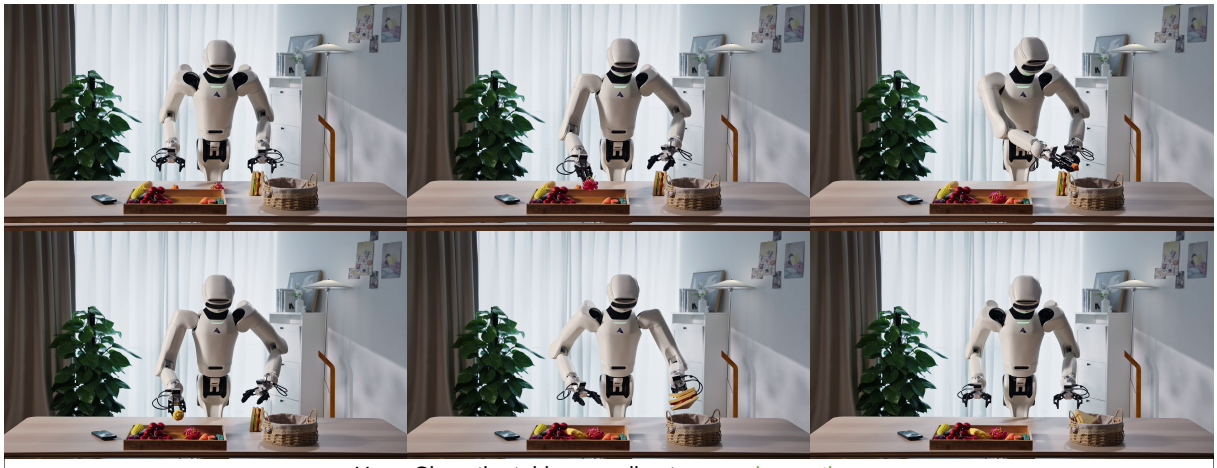
To balance accuracy and latency, we adopt a five-stage cascaded architecture tailored to the heavy-tailed distribution of intent complexity, with latency breakdowns in Tab. 2. Ordered by increasing complexity: (1) regex / deterministic fast-pass, (2) local fast classifier, (3) deterministic semantic router, (4) capability-grounded semantic router, (5) end-to-end agent execution.

Tier	p50 (ms)	p90 (ms)	p99 (ms)	N
1. Regex / deterministic fast-pass	1	1	1	5
2. Local fast classifier (F1 0.965)	414	419	422	483
3. Deterministic semantic router	1	1	1	5
4. Capability-grounded semantic router	3 609	4 433	6 197	483
5. Agent turn, non-robot	4 857	17 484	52 327	2014
5. Agent turn, robot tool	25 348	57 264	288 890	122

Table 2 | **Five-tier dispatch latency.** Tiers handle intents of increasing complexity; most traffic resolves before the heavyweight grounded router.

This hierarchical design reduces average inference cost while preserving high dispatch accuracy: most simple intents are resolved at low-latency early tiers, and only complex ambiguous requests reach the heavyweight router. Our deployed configuration uses Qwen3.5-4B for pre-filtering and claude-opus-4.6 for full routing, achieving an optimal precision-latency tradeoff: 100% non-robot rejection at 3.6 s p50 routing latency.

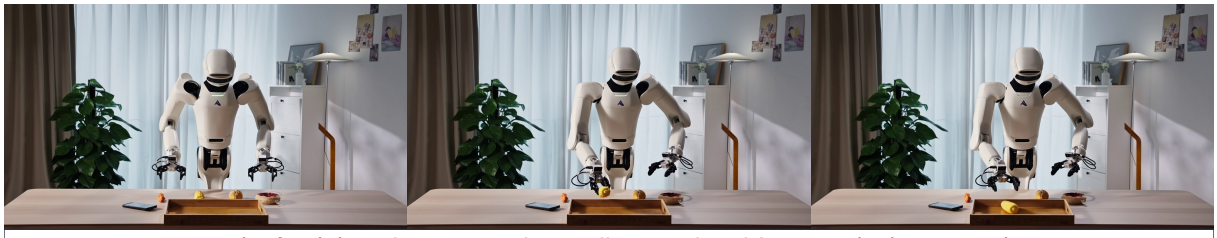
(a) Observation-grounded reasoning.



User: Clean the table according to your observation.

Reasoning: The large wooden tray already holds vegetables and fruit (carrots, lychees, eggplant, corn), so I placed the loose grape and corn there since grapes are fruit and corn is a vegetable. The round woven basket already contains bread and pastries, so I placed the fried chicken leg and the sandwich there because they all belong to the same food category.

(b) Preference-aware reasoning.



User: Put the food that Chinese people usually eat at breakfast into the large wooden tray.

Reasoning: Corn is one of the breakfast foods commonly eaten by Chinese people(such as boiled corn, corn porridge), so I chose the corn visible on the table.

(c) High-calorie reasoning.



User: Put the high-calorie food into the large wooden tray.

Reasoning: The donut and fried chicken leg are high-calorie foods on the table, so I placed them into the large wooden tray.

(d) Food-category reasoning.



User: Put all the vegetables into the large wooden tray.

Reasoning: The corn and the plastic carrot are vegetables on the table and I placed them into the wooden tray. No loose vegetables remain on the table — the remaining loose objects are a alpaca, a donut, and a small animal figure, none of which are vegetables. The sweep is complete.

Figure 2 | **Reasoning-grounded action.** PHILIA transfers the agent's reasoning capability into policy execution. The examples demonstrate grounding in observations, user preferences, calorie knowledge, and food-category semantics.

3.3. PHILIA Playbook

We validate PHILIA through representative deployment scenarios on Astribot S1 robots. Each scenario exercises a distinct system capability: reasoning-grounded action, memory-grounded action, multi-robot coordination, or plug-and-play policy execution.

Agent-to-Policy Reasoning Grounding. PHILIA grounds high-level agent reasoning in low-level policy execution by constructing policy prompts from the user instruction, the robot's observation, and relevant external knowledge. As shown in Fig. 2a, the agent combines scene observations with pretrained knowledge and web-search results before invoking a policy skill. This mechanism provides open-vocabulary grounding for policy execution and enables the robot to act on semantic constraints that are not hard-coded in the policy itself. Additional examples in Fig. 2b, Fig. 2c, and Fig. 2d show reasoning grounded in user preferences, calorie knowledge, and food-category semantics.

Agent-to-Policy Long-Term Memory Grounding. PHILIA also grounds physical action execution in long-term user memory. As shown in Fig. 3, the user previously told PHILIA that he usually has sandwiches and espresso for breakfast. PHILIA stores this preference in memory and later retrieves it as context when handling related robot tasks.



Figure 3 | **Memory-grounded action.** PHILIA retrieves long-term user preferences to guide related robot task execution.

One Agent, Multi-Robot Control. PHILIA represents each robot through the same robot-gateway runtime abstraction. Adding a new robot therefore requires only instantiating a gateway and registering it with PHILIA. As shown in Fig. 4, PHILIA can control multiple robots by assigning tasks to their respective gateways in parallel, enabling robot Alice to clean up the table while robot Bob lifts the garbage bag.

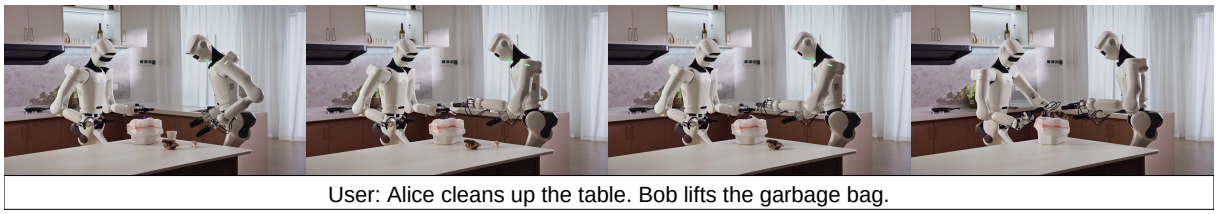


Figure 4 | **One Agent, Multi-Robot Control.** PHILIA controls multiple robots through independently registered robot gateways.

Plug-and-Play Policy Execution. PHILIA treats each policy as a registered module, enabling plug-and-play policy execution across robots. Fig. 5 shows PHILIA invoking the Lumo-2 pack-the-backpack policy while preserving the robot's execution accuracy.

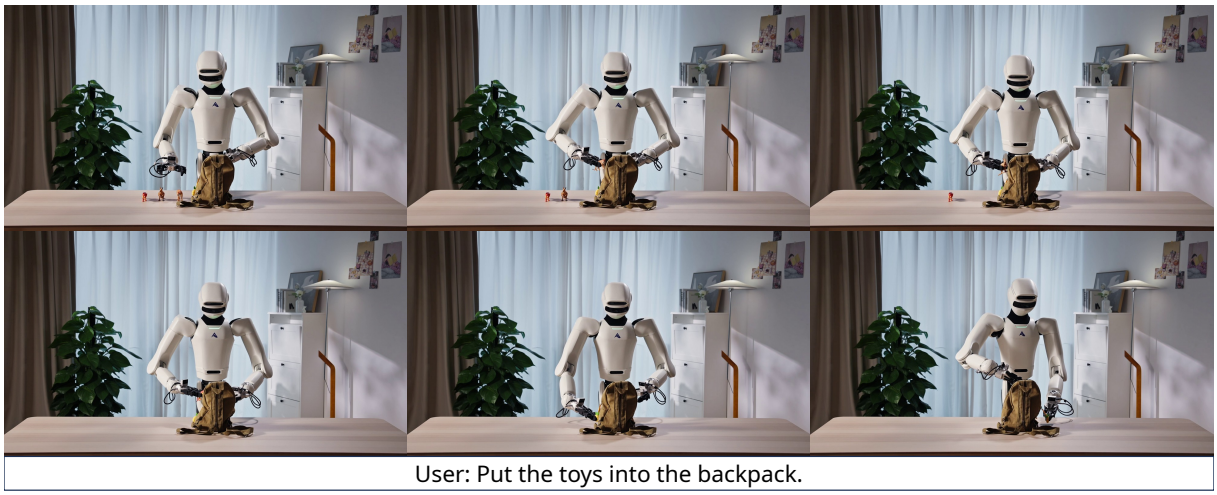


Figure 5 | **Plug-and-play policy execution.** PHILIA invokes a registered pack-the-backpack policy on the Lumo-2 robot.

4. Related Work

LLM-Based Robot Agents and Generalist Policies. Language-model-based robot systems have explored grounding instructions in affordances, tool calls, code, multi-modal representations, and spatial value maps, demonstrating that large models can provide high-level reasoning for embodied tasks (Ahn et al., 2022, 2024; Driess et al., 2023; Huang et al., 2023; Liang et al., 2023). More recent work extends these ideas to multi-robot task allocation, role assignment, and long-horizon collaboration (Liu et al., 2023; Sarkar et al., 2025; Zhang et al., 2025). In parallel, large-scale robot learning has produced generalist policies that connect language, vision, and action across robot embodiments and datasets (Black et al., 2024; Brohan et al., 2022; Kim et al., 2024; O'Neill et al., 2024; Team et al., 2024; Zitkovich et al., 2023), with recent work further exploring reasoning-aware action models, bimanual manipulation, and whole-body mobile manipulation (Fu et al., 2024; Liu et al., 2024; Tang et al., 2025; Zawalski et al., 2024; Zhao et al., 2025). PHILIA is complementary to these advances. Rather than proposing a new policy or asking the language model to directly control robots, it treats policies and robot skills as replaceable robot-local execution backends behind a shared gateway contract. The assistant remains responsible for intent interpretation, actor resolution, semantic memory, human-facing interaction, and recovery-level reasoning.

Long-Term Human–Robot Coexistence, Memory, and Personalization. Long-term service robots must operate in changing environments, preserve useful experience, and remain understandable to human users. The STRANDS project demonstrated extended autonomous deployments in everyday environments and highlighted the need to combine navigation, perception, interaction, planning, and learning for persistent service robots (Hawes et al., 2017; Kunze and Hawes, 2018). Continual-learning studies for robotics and home service robots further show that personalization requires repeated human interaction, semantic knowledge

updates, and careful handling of forgetting and trust (Ayub et al., 2024; Lesort et al., 2020). Personalized assistants such as TidyBot demonstrate how language models can infer household preferences from experience (Wu et al., 2023). At the same time, modern mobile robots rely on mature mapping, localization, and navigation systems, including LiDAR-inertial SLAM, graph optimization, semantic mapping, and planner stacks (Hess et al., 2016; Kavraki et al., 1996; Kostavelis and Gasteratos, 2015; Kümmerle et al., 2011; Macenski et al., 2020; Shan et al., 2020; Thrun et al., 2005; Xu et al., 2022). PHILIA treats persistent memory and robot geometry as shared support infrastructure for long-term coexistence: semantic memory informs planning across users and robots, while each robot retains its own maps, localization, and navigation stack.

Runtime Abstractions for Robot Execution and Recovery. Recent systems increasingly treat robots as components of an agentic runtime rather than isolated policies. RoboClaw-style systems expose robot services and skills through assistant-facing interfaces (Spin-Matrix, 2026), RoboClaw integrates data collection, policy learning, execution monitoring, and recovery (Li et al., 2026), while RoboAgent studies long-horizon capability composition in simulated environments (Xu et al., 2026). ABot-Claw further extends OpenClaw toward persistent cooperative robotic agents, combining a unified embodiment interface, cross-embodiment multimodal memory, and critic-based closed-loop feedback for heterogeneous robot coordination (Huo et al., 2026). Robotic systems also often structure task execution with modular control abstractions such as behavior trees, state machines, and task-level monitors; behavior trees are widely used because they are reactive, modular, and analyzable for robot missions (Colledanchise and Ögren, 2018; Ghzouli et al., 2023; Iovino et al., 2022). PHILIA similarly views robot execution as more than a single policy invocation, but introduces a deployment-oriented control plane over heterogeneous robot actors. Platform-specific execution, safety, monitoring, and recovery remain robot-local responsibilities exposed through gateway capabilities and status signals, while the assistant observes progress, requests cancellation, asks for human confirmation, and triggers bounded recovery without assuming authority over continuous robot control.

Middleware and Heterogeneous Robot Integration. Robotic middleware such as ROS and ROS 2 provides composition of drivers, sensors, planners, and policies within a single robot (Macenski et al., 2022; Quigley et al., 2009). PHILIA operates at a different layer. It assumes each robot already possesses a functioning middleware-level software stack and exposes only a narrow semantic interface through a gateway. This allows robots built on ROS, ROS 2, or proprietary software to share a common assistant-facing capability contract. Rather than replacing planners, policies, navigation systems, or interaction interfaces, PHILIA provides the deployment abstraction that connects them, enabling heterogeneous robot embodiments to be orchestrated as persistent physical agents behind a unified assistant identity. This separation makes actor-scoped routing, backend binding, capability discovery, stop/cancel ownership, and plug-in robot backends first-class system abstractions rather than implementation details.

5. Discussion

5.1. Core Design Tradeoffs

The architecture is built around two deliberate tradeoffs that shape all higher-level design decisions:

1. **Stable boundary vs. Granular state access.** PHILIA prioritizes a narrow, stable gateway interface over full exposure of robot internals. This enables composable evolution across layers, but limits the agent's access to low-level execution state. The agent's ability to recover from failures (ambiguous scenes, blocked navigation, policy errors) depends entirely on the quality of semantic summaries published through the gateway contract.
2. **Semantic flexibility vs. Physical safety.** The agent retains full flexibility for intent understanding, preference reasoning and task decomposition, but all state-changing physical actions pass through actor-scoped capabilities and safety gates. This preserves natural interaction richness without granting the language model direct authority over high-frequency motion control.

Consistent with this boundary design, PHILIA does not aim to replace underlying VLA policies, navigation stacks or robot middleware; instead, it provides a unifying runtime contract to compose them under a single assistant identity. This modular, boundary-driven design also determines the system's evolution pattern: user experience improves incrementally via upgrades to any individual layer (UI, policy, navigation, memory, dialogue), rather than relying on monolithic end-to-end model replacement. The gateway contract must therefore balance expressiveness and long-term stability.

5.2. Heterogeneous Platform Support and Security

The architecture natively supports integration of heterogeneous robot platforms beyond the currently validated Astribot S1. New robot models (e.g., Astribot T1) can be added by implementing a gateway adapter and publishing a capability manifest, with no changes to the agent control plane. This design property is not yet experimentally validated on non-S1 platforms, and remains a forward-looking feature of the architecture.

For physical security, authorization, confirmation, active-task arbitration and actor-scoped stop/cancel are built into the core system, not treated as optional UI features. The gateway boundary restricts the agent to structured capability requests, with platform credentials, map data and low-level safety checks retained exclusively in robot-local runtimes, narrowing the authority surface for physical execution.

5.3. Limitations and Future Work

This work focuses on architectural design and representative real-world use cases, not large-scale controlled benchmarking. The end-to-end task performance of manipulation, navigation and alignment depends on the quality of the underlying robot-local backends, which are treated as pluggable components.

Key open directions for future work include: quantifying long-term reliability and failure recovery rates in sustained daily operation; evaluating multi-user authorization and permission management; and validating cross-platform generalization across heterogeneous robot embodiments beyond S1. We also note that current gateway semantics remain relatively coarse-grained, and finer-grained state feedback could further improve agent-level failure handling.

6. Conclusion

We presented PHILIA, a system architecture for long-term physical coexistence with intelligent robots. The central idea is to keep a persistent assistant identity at the semantic control plane while exposing robot platforms through simple robot gateway abstractions. Rather than replacing underlying VLA policies, navigation stacks or robot middleware, PHILIA provides a unifying runtime contract to compose them under a single assistant identity. This design preserves rich interaction, tools, UI surfaces, memory, and planning from OpenClaw, while delegating fine-tuned policies, navigation, sensing, and safety-critical execution to robot-local runtimes.

New robot platforms can be integrated by implementing a gateway adapter and publishing a capability manifest, with no modifications to the agent control plane. Through representative use cases, PHILIA illustrates how agent-level reasoning and long-horizon context can be grounded into physical robot action without tightly coupling the assistant to a single body, robot model, or policy backend.

Instead of replacing human agency, judgment and responsibility, systems like PHILIA extend human capability by distributing cognition across people, machines and environments. This work is an invitation to explore how intelligent robots can become reliable, personalized partners in long-term daily coexistence.

7. Contributions

Author contributions in the following areas are listed in alphabetical order.

- **System Design:** Weiqi Jin
- **Contributors:** Baifu Huang, Binyan Sun, Haotian Yang, Kuncheng Luo, Peijun Tang, Shangjin Xie
- **Project Lead:** Jianan Wang

References

M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan. Do as i can, not as i say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691, 2022.

M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, E. Jang, N. Joshi, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, D. Sadigh, P. Sermanet, F. Xia, T. Xiao, P. Xu, S. Xu, et al. Autort: Embodied foundation models for large scale orchestration of robotic agents. arXiv preprint arXiv:2401.12963, 2024.

Anthropic. Claude models overview. `https://platform.claude.com/docs/en/about-claude/models/overview`, 2026. Accessed: 2026-07-11.

A. Ayub, C. L. Nehaniv, and K. Dautenhahn. An interactive continual learning architecture for long-term personalization of home service robots. arXiv preprint arXiv:2403.02012, 2024.

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164, 2024.

K. Black, M. Y. Galliker, and S. Levine. Real-time execution of action chunking flow policies, 2025a. URL `https://arxiv.org/abs/2506.07339`.

K. Black, A. Z. Ren, M. Equi, and S. Levine. Training-time action conditioning for efficient real-time chunking, 2025b. URL `https://arxiv.org/abs/2512.05964`.

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

M. Colledanchise and P. Ögren. Behavior Trees in Robotics and AI: An Introduction. CRC Press, 2018.

D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, et al. Palm-e: An embodied multimodal language model. 2023.

Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. arXiv preprint arXiv:2401.02117, 2024.

G. Gao, J. Wang, J. Zuo, J. Jiang, J. Zhang, X. Zeng, Y. Zhu, L. Ma, K. Chen, M. Sheng, R. Zhang, and Z. An. Towards human-level intelligence via human-like whole-body manipulation, 2025. URL `https://arxiv.org/abs/2507.17141`.

R. Ghzouli, T. Berger, E. B. Johnsen, A. Wasowski, and S. Dragule. Behavior trees in action: A study of robotics applications. IEEE Robotics & Automation Magazine, 30(2):90–103, 2023.

Google. Gemini 3.5 Flash. `https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash`, 2026. Accessed: 2026-07-11.

N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrová, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, et al. The STRANDS Project: Long-term autonomy in everyday environments. IEEE Robotics & Automation Magazine, 24(3):146–156, 2017.

W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2D LIDAR SLAM. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 1271–1278. IEEE, 2016.

W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. arXiv preprint arXiv:2307.05973, 2023.

D. Huo, H. Liu, G. Liu, D. Qi, Z. Sun, M. Gao, J. He, Y. Yang, X. Chang, F. Xiong, X. Wei, Z. Ma, and M. Xu. ABot-Claw: A foundation for persistent, cooperative, and self-evolving robotic agents, 2026. URL `https://arxiv.org/abs/2604.10096`.

P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian, K. Dhabalia, J. DiCarlo, D. Driess, M. Equi, A. Esmail, Y. Fang, C. Finn, C. Glossop, T. Godden, I. Goryachev, L. Groom, H. Hancock, K. Hausman, G. Hussein, B. Ichter, S. Jakubczak, R. Jen, T. Jones, B. Katz, L. Ke, C. Kuchi, M. Lamb, D. LeBlanc, S. Levine, A. Li-Bell, Y. Lu, V. Mano, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, C. Sharma, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, W. Stoeckle, A. Swerdlow, J. Tanner, M. Torne, Q. Vuong, A. Walling, H. Wang, B. Williams, S. Yoo, L. Yu, U. Zhilinsky, and Z. Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL `https://arxiv.org/abs/2511.14759`.

M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith. A survey of behavior trees in robotics and ai. Robotics and Autonomous Systems, 154:104096, 2022.

L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4):566–580, 1996.

M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts, 2019. URL `https://arxiv.org/abs/1810.02890`.

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

I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. In Robotics and Autonomous Systems, volume 66, pages 86–103, 2015.

R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A general framework for graph optimization. In 2011 IEEE International Conference on Robotics and Automation, pages 3607–3613. IEEE, 2011.

L. Kunze and N. Hawes. Long-term autonomy and artificial intelligence for mobile robots. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

I. Larchenko, G. Zarin, and A. Karnatak. Task adaptation of vision-language-action model: 1st place solution for the 2025 behavior challenge. arXiv preprint arXiv:2512.06951, 2025.

T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. Information Fusion, 58:52–68, 2020.

R. Li, Y. Zhou, Y. Zhu, K. Chen, J. Wang, S. Wang, K. Hu, M. Yu, B. Jiang, Z. Su, J. Ma, X. He, Y. Shen, Yangyang, G. Ren, M. Yao, W. Wang, and Y. Mu. Roboclaw: An agentic framework for scalable long-horizon robotic tasks, 2026. URL https://arxiv.org/abs/2603.11558.

J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. arXiv preprint arXiv:2209.07753, 2023.

B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Smart-llm: Smart multi-agent robot task planning using large language models. arXiv preprint arXiv:2309.10062, 2023.

S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. arXiv preprint arXiv:2410.07864, 2024.

S. Macenski, F. Martín, R. White, and J. G. Clávero. The marathon 2: A navigation system. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2718–2725, 2020.

S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics, 7(66):eabm6074, 2022.

OpenAI. GPT-5.4 model documentation. https://developers.openai.com/api/docs/models/gpt-5.4, 2026a. Accessed: 2026-07-11.

OpenAI. GPT-5.4 mini model documentation. https://developers.openai.com/api/docs/models/gpt-5.4-mini, 2026b. Accessed: 2026-07-11.

OpenAI. GPT-5.5 model documentation. https://developers.openai.com/api/docs/models/gpt-5.5, 2026c. Accessed: 2026-07-11.

A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 6892–6903. IEEE, 2024.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: An open-source robot operating system. In ICRA Workshop on Open Source Software, volume 3, page 5. Kobe, Japan, 2009.

Qwen Team. Qwen3.5-4B model card. https://huggingface.co/Qwen/Qwen3.5-4B, 2026. Accessed: 2026-07-11.

B. Sarkar, A. Pal, R. Hazra, T. Gokhale, C. Baral, Y. Yang, and Z. Kira. Llamar: Long-horizon planning for multi-agent robots in partially observable environments. arXiv preprint arXiv:2501.09743, 2025.

T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5135–5142. IEEE, 2020.

Spin-Matrix. Roboclaw: A lightweight intelligent control project for robot scenarios. https://github.com/spin-matrix/roboclaw, 2026. GitHub repository.

P. Steinberger and O. Contributors. Openclaw: Your own personal ai assistant. https://github.com/openclaw/openclaw, 2026. Accessed: 2026-07-04.

P. Tang, S. Xie, B. Sun, B. Huang, K. Luo, H. Yang, W. Jin, and J. Wang. Mind to hand: Purposeful robotic control via embodied reasoning. arXiv preprint arXiv:2512.08580, 2025.

O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. arXiv preprint arXiv:2405.12213, 2024.

S. Thrun, W. Burgard, and D. Fox. Probabilistic robotics. Communications of the ACM, 45(3): 52–57, 2005.

J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. Tidybot: Personalized robot assistance with large language models. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3546–3553. IEEE, 2023. doi: 10.1109/IROS55552.2023.10341577.

P. Xu, J. Zheng, and Y. Mu. Roboagent: Chaining basic capabilities for embodied task planning, 2026. URL `https://arxiv.org/abs/2604.07774`.

W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. FAST-LIO2: Fast direct lidar-inertial odometry. IEEE Transactions on Robotics, 38(4):2053–2073, 2022. doi: 10.1109/TRO.2022.3141876.

C. Yu, C. Sima, G. Jiang, H. Zhang, H. Mai, H. Li, H. Wang, J. Chen, K. Wu, L. Chen, L. Zhao, M. Shi, P. Luo, Q. Bu, S. Peng, T. Li, and Y. Yuan. χ_0: Resource-aware robust manipulation via taming distributional inconsistencies, 2026. URL `https://arxiv.org/abs/2602.09021`.

M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. arXiv preprint arXiv:2407.08693, 2024.

Y. Zhang, G. Chen, J. Li, Z. Wang, X. Yang, and W. Huang. Large language models for multi-robot systems: A survey. arXiv preprint arXiv:2502.03814, 2025.

Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 1702–1713, 2025.

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, pages 2165–2183. PMLR, 2023.