

Adaptive Code Generation for Controlling Robots

Justus Flerlage¹, Thorsten Wittkopp^{1,2}, Alexander Acker², and Odej Kao¹

¹ Technische Universität Berlin, Germany

{j.flerlage, t.wittkopp, odej.kao}@tu-berlin.de

² logsight.ai, Berlin, Germany

{thorsten.wittkopp, alexander.acker}@logsight.ai

Abstract. Deploying robots as Complex Adaptive Systems (CAS) in unknown and dynamic environments necessitates a transition from rigid command libraries toward intention-based autonomy, as natural language represents the only medium capable of articulating complex goals beyond the capacity of finite instruction sets. While Large Language Models (LLMs) offer a path toward natural language goal description, their integration introduces significant challenges: the formalization gap between imprecise intentions and executable actions, the taxonomy gap induced by unpredictable environments, and the challenge of maintaining temporal state and progress awareness. This work introduces an architectural framework that enables robotic control by leveraging generative AI. The system follows a dual-AI design: an LLM translates high-level intentions into executable program code restricted to a formal robotic library and constrained by verifiable syntax, while a Vision-Language Model (VLM) provides semantic grounding via a distillation process. To ensure robustness, the framework incorporates environment-driven replanning triggers based on geometric and semantic thresholds, complemented by continuous runtime monitoring and an adaptive planning loop. Benchmarked across frontier models, our framework architecture demonstrates that grounding generative AI in a reactive, constrained loop enables robust fulfillment of complex intentions in dynamic and unknown environments.

Keywords: Autonomous Robots · Adaptive Code Generation · Large Language Models · Artificial Intelligence.

1 Introduction

CAS are designed to operate under uncertainty, partial knowledge, and continuous environmental change. Rather than relying on a single centralized controller with a complete model of the world, CAS coordinates heterogeneous components whose collective behavior emerges through feedback, adaptation, and interaction with a dynamic environment [24,1]. Robotic systems operating outside static industrial settings constitute a particularly challenging instance of this paradigm. They must interpret high-level goals, perceive incomplete and changing environments, select appropriate actions, and revise their behavior when assumptions about the world become invalid.

However, conventional robotic systems are typically designed for static and well-defined environments and tasks. The prevailing paradigms of conventional robotics have historically centered on low-level control, symbolic planning, and rule-based systems [13,20]. Within industrial environments, such as automotive assembly lines, robots execute predetermined operations based on a closed-world assumption, utilizing deterministic algorithms, fixed rule sets, and task models that are entirely specified in advance. Even in domains where Artificial Intelligence (AI) appears to provide greater flexibility, such as autonomous driving or industrial manipulation, system behavior remains constrained by predefined environmental assumptions [11,16]. A notable illustration of this is autonomous driving, wherein AI models are trained to recognize a specific set of object classes, such as pedestrians, traffic signs, and lane markings [11]. While such approaches are demonstrably effective within their intended domains, they remain inherently limited in scope due to their reliance on predefined taxonomies and fixed execution models. To provide an example, an autonomous car may navigate safely through a city, but it may fail to respond appropriately when a traffic officer overrides traffic signs or signals in a manner not represented in the vehicle’s training data. Consequently, they are unable to generalize to dynamic and open-ended environments in which robots encounter novel objects, conditions, and unknown tasks [7].

This leads to the first challenge in the field of adaptive robotics, where robots are expected to operate in environments and task settings that cannot be fully specified at design time. Addressing this challenge requires a shift toward natural language as a versatile medium for task specification, enabling robots to interpret and execute a virtually limitless range of complex instructions that cannot be captured by rigid, predefined taxonomies. The inherent flexibility of natural language introduces a significant secondary challenge: such high-level intentions are frequently underspecified, leaving a wide semantic gap between a human’s intent and the granular actions a robot must perform. To illustrate this point, one may consider a scenario in which a robot is assigned the intention of requesting a coffee. It is conceivable that such tasks could encompass the instruction of a robot to explore an unknown environment, inspect a room, search for relevant objects, operate a machine, and retrieve a requested coffee. However, this instruction does not include several key elements. Firstly, the text provides no indication as to which objects the robot is to encounter and how to react to them. Secondly, the text fails to specify which areas the robot should visit first. Thirdly, the text does not provide a definitive timeframe within which the exploration should be deemed complete. In conclusion, translating a simple human intention into a concrete operational plan remains a major hurdle, as the robot must autonomously infer the missing procedural details and environmental constraints.

Recent advances in generative AI offer a promising foundation for this type of intention-driven adaptation. LLMs have the capacity to interpret natural language instructions, infer task goals, generate action plans, and reason about task progress [6,2,17,18]. Nevertheless, intention-driven robotic control also ne-

cessitates the grounding of these plans in the physical environment. To achieve this, VLMs provide a mechanism for deriving semantic descriptions from visual observations, enabling robots to reason about objects and spatial relations that extend beyond predefined categories [19]. The integration of generative AI into robotic control systems introduces a fundamental tension, as the properties that render generative models attractive for open-ended adaptation also render them difficult to integrate into reliable control loops. It is evident that both LLMs and VLMs possess a degree of flexibility that enables them to function on instructions that are underspecified and on incomplete scene information. However, it is important to acknowledge that the outputs of these systems are non-deterministic and not guaranteed to be correct. Furthermore, these outputs are based on continuously evolving representations of the environment. For CAS, this necessitates a framework where generative AI is embedded into an adaptive feedback architecture without compromising reliability. This challenge motivates the following research question:

How can generative AI be integrated into a Collective Adaptive System for robotic control such that user intentions can be translated into reliable robot behavior under uncertainty, incomplete knowledge, and continuous environmental change?

The question is addressed by means of an architectural framework for intention-driven robotic task execution in dynamic and unstructured environments. The architectural framework implements the MAPE-K feedback loop, meticulously decomposing the process of robotic adaptation into five distinct phases: monitoring, analysis, planning, execution, and knowledge management [4]. A VLM-based perception component is responsible for monitoring the environment and producing open-ended semantic descriptions of the robot’s surroundings. A planning component, based on an LLM, is capable of interpreting user intentions and generating executable robot behavior. A library of robotic control functions constrains the LLM’s output to valid actions, thereby enabling syntactic verification before execution. A workflow-monitoring and knowledge component maintains information about execution progress, previous actions, and observed scene changes, thereby supporting adaptation over long-running tasks.

Our architectural framework addresses three fundamental challenges in the domain of intention-driven adaptive robotics. Firstly, it mitigates the formalization gap between imprecise natural language intentions and the precise control structures required for robotic execution. Our framework employs an advanced LLM that translates user intentions into workflows. Secondly, it addresses the taxonomy gap that arises when robots encounter objects and situations that cannot be fully anticipated at design time. Rather than relying exclusively on a fixed object taxonomy, the VLM provides open-ended semantic grounding, which is stabilized over time through repeated observations. Thirdly, it addresses the temporal state problem of generative controllers by maintaining explicit execution state, thus allowing the system to distinguish between completed subtasks, failed actions, and pending objectives.

In order to investigate the aforementioned research question, a proof-of-concept implementation of the proposed architecture for a quadruped robot performing exploration tasks is presented. The robot receives high-level user intentions rather than explicit programs and must autonomously generate, execute, and adapt task behavior in response to its environment. The system is evaluated across a range of object-identification and exploration tasks and scenarios, utilizing both finite and open-ended tasks. Furthermore, a range of LLMs are being benchmarked to ascertain how variations in generative models impact task success, robustness, replanning behavior, and termination control.

The following contributions are made by the paper:

1. The present paper proposes a MAPE-K-inspired architectural framework for intention-driven robotic control in dynamic and unstructured environments.
2. The integration of LLMs and VLMs into a constrained adaptive feedback loop that combines semantic grounding, executable code generation, monitoring, and persistent task knowledge is demonstrated.
3. The performance of three state-of-the-art LLMs is evaluated as a fundamental component for intention-driven robotic control in dynamic and unstructured environments, utilizing exploration tasks that necessitate spatial and temporal reasoning, executable code generation, programming capabilities, and mathematical problem-solving.

In Section 2 the framework architecture is presented. While Section 3 describes the methods and implementation, Section 4 shows an evaluation. After Section 5 addresses the related work, Section 6 concludes our work.

2 Framework Architecture

The proposed framework architecture is founded on the principles of MAPE-K, a standard architectural pattern in CAS [4]. This loop, comprising Monitoring, Analysis, Planning, and Execution, operates over a shared knowledge base that functions as a stigmergic medium. As illustrated in Figure 1, the framework architecture is divided into a perception component, a cognition component, and a centralized world state. Rather than serving as a passive repository, this world state provides the environmental cues and geometric grounding necessary to drive adaptation through the environment itself. The framework architecture is specifically engineered to address the inherent complexities of autonomous robotic adaptation by targeting three critical challenges: the taxonomy gap, formalization gap, and the temporal state problem.

The perception component mitigates the taxonomy gap, the discrepancy between a robot’s fixed internal categories and the infinite variety of the physical world, by decoupling geometric grounding from semantic identification. Instead of relying on closed-set classifiers, the system utilizes a VLM to perform open-ended semantic enrichment of raw sensor data. This information is synchronized with the world state, which functions not as a passive repository but as a stigmergic medium that provides the environmental cues necessary for cognitive reasoning. By grounding abstract semantic labels in precise geometric coordinates,

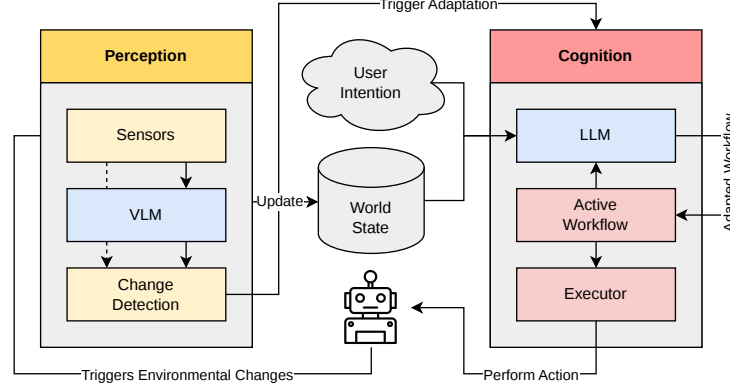


Fig. 1. MAPE-K influenced framework architecture for autonomous robots resolving user intentions in dynamic and unstructured environments.

the system ensures that the robot can interact with and reason about previously unidentified objects without requiring a predefined taxonomic library.

Bridging the formalization gap between high-level user intentions and low-level robotic control requires a rigorous approach. Within the cognition component of this framework architecture, processes are organized around workflows modeled as Turing-complete imperative programs. This structure allows the system to map ambiguous natural language intentions onto a formal sequence of actions restricted to a predefined library of high-level robotic primitives for movement control, ensuring both safety and controllability. The LLM acts as the central decision-making entity within this planning phase, analyzing the current world state and the active workflow to generate updated logic in response to environmental triggers. This transformation of intent into executable code provides a formal bridge between the non-deterministic nature of human language and the deterministic requirements of robotic execution.

To maintain operational continuity during these adaptations, the framework architecture explicitly addresses the temporal state problem, which occurs when a system loses track of its execution progress upon receiving a new plan. The executor facilitates this by utilizing a virtual machine capable of capturing snapshots of the internal execution state, including instruction pointers and variable bindings. When a change in the environment triggers a re-planning event, the current state is preserved and provided to the LLM, enabling the synthesis of an adapted workflow that can be hot-reloaded mid-execution. This approach ensures that the robot does not reset its task upon every environmental shift, but rather adapts its behavior while maintaining full state awareness. Consequently, the entire MAPE-K loop operates as an event-driven paradigm where the system continuously perceives its surroundings, reasons about semantic and geometric changes, and executes formal, state-aware adaptations in real-time.

3 Methods and Implementation

The realization of the framework architecture described in Section 2 requires methods to ground environmental data and formalize the adaptation loop. This section details the algorithmic approach to perception, the mathematical definition of adaptation triggers, and the implementation of the perception and cognition components.

3.1 Perception Component

The perception layer converts raw sensor streams into the stigmergic world state. This process relies on geometric clustering to identify physical entities and asynchronous vision-language inference for semantic enrichment. Even in static settings, the internal model remains highly dynamic due to the incremental nature of discovery. An object’s full semantic and spatial profile is rarely visible from a single perspective. To resolve this, the component constantly fuses heterogeneous sensor data, while the robot is exploring the world: 2D LiDAR provides geometric dimensions and distances, while a VLM provides unrestricted natural language descriptions. This fusion allows the system to manage a shifting world state, where initial geometric hypotheses are continuously refined by new viewing angles and semantic updates.

Geometric Grounding and AABB Logic To bridge the gap between raw, unstructured sensor data and the higher-level semantic understanding required by the LLM, we must transform LiDAR observations into stable world observations. Therefore, physical entities are represented as Axis-Aligned Bounding Boxes (AABB) in the Cartesian plane. To transform unstructured LiDAR point clouds into entities within the perception layer, the system utilizes a sequential clustering algorithm that maintains spatial consistency by accounting for the decrease in ray density. For any two sequential points p_i and p_{i-1} located at radial distances d_i and d_{i-1} from the sensor, the dynamic distance threshold δ_{arc} is defined as:

$$\delta_{arc} = C \cdot \max(d_i, d_{i-1}) \cdot \theta_{res} \quad (1)$$

where θ_{res} represents the angular resolution in radians, and C is a constant multiplier acting as a tolerance buffer for sensor noise. Points are clustered if their Euclidean distance is less than δ_{arc} . Once a set of points $\mathcal{P} = \{p_1, \dots, p_n\}$ is assigned to a cluster, the system defines the entity as an Axis-Aligned Bounding Box whose boundaries are the minimum and maximum coordinates of the point set $\mathcal{P}$ along each global axis:

$$x_{min} = \min_{p \in \mathcal{P}} x_p, \quad x_{max} = \max_{p \in \mathcal{P}} x_p, \quad y_{min} = \min_{p \in \mathcal{P}} y_p, \quad y_{max} = \max_{p \in \mathcal{P}} y_p \quad (2)$$

To ensure the robustness of the world state, clusters are only promoted to formal AABB entities if they exceed a density threshold N_{min} , defined as 1% of

the total LiDAR ray count. Temporal stability of AABB entities over time is maintained through an Exponential Moving Average (EMA) with a distance-dependent learning rate $\alpha = \exp(-d)$, requiring the world state to be updated with every LiDAR measurement as new spatial evidence is gathered. This continuous refinement is necessary to mitigate sensor imprecision and consolidate varying perspectives as the robot views objects from different angles.

Asynchronous Semantic Enrichment Semantic data is integrated into the world state through a Vision-Language Model (VLM). When an AABB enters the camera’s field of view, the system triggers an asynchronous inference request using a centered RGB image. The VLM estimates the object’s physical footprint and provides natural language tags. These tags are subsequently fused into the World State by performing a spatial join between the VLM estimated AABB and the existing LiDAR derived hitboxes. To mitigate transient hallucinations, the system maintains a running frequency distribution of tags for each entity. Only the top- k most frequent tags are surfaced to the cognition component, ensuring that semantic grounding remains statistically stable across multiple frames.

3.2 Event-Driven Adaptation Triggers

The transition from monitoring to planning, the Analysis phase of the MAPE-K loop, is governed by three mathematical triggers. These triggers serve as gatekeepers to ensure the Large Language Model (LLM) is invoked only when environmental shifts exceed significant thresholds. The **New Object Trigger** is activated immediately upon the promotion of a new AABB from a raw cluster to a formal entity in the world state. The **Geometric Change Trigger** monitors the historical area A of active AABBs. A re-planning request is issued if the current area A_t satisfies the growth condition $\frac{A_t}{A_{initial}} > \gamma$, where $A_{initial}$ is the first stable area in the history buffer and γ is a predefined growth threshold. This captures significant discoveries, such as the revelation of an occluded surface. Finally, the **Semantic Discovery Trigger** is activated if the VLM inference results in a shift where n out of the k most frequent tags change, indicating that the understood nature of an entity has fundamentally evolved.

3.3 Cognition Component

The cognition component functions as the event-driven loop mechanism performing monitoring, analyzing, planning, and execution using an LLM and the executor by incorporating an active workflow, a user intention and adaptation triggers.

A workflow is formally represented as a program written with a Turing-complete imperative programming language. For the purposes of this implementation, a workflow is represented as a Python 3 program, based on the assumption that LLMs are well-trained on publicly available data and Python 3 is one of the most popular programming languages according to the Tiobe index [12]. The

executor comprises a source-to-bytecode compiler utilizing the default Python 3 implementation, CPython, and a custom stack-based immutable virtual machine for executing Python 3 bytecode.

The virtual machine provides isolation and controlled execution, whilst also supporting granular hot code reloading. This enables dynamic updates to workflows without compromising runtime operation. In order to facilitate adaptation, the virtual machine creates snapshots of its internal state. These snapshots include the current instruction, variable bindings, function parameters and call stack. This facilitates seamless recovery and modification, thereby ensuring robust and fault-tolerant execution in dynamic environments.

4 Evaluation

In order to validate the proposed framework architecture, the system is evaluated on a benchmarking suite specifically designed for quadruped robot exploration tasks. The benchmarks target the mitigation of taxonomy, formalization, and temporal state gaps by testing the robot’s ability to interpret high-level exploration goals, maintain consistent state representations over time, and execute adaptive behaviors in partially observable environments. This setting facilitates an evaluation of the architecture’s capacity to facilitate effective decision-making processes for embodied agents operating within complex and dynamically evolving terrain.

4.1 Experimental Setup

We evaluate our system using Gazebo as a simulation environment, which communicates with the aforementioned components using the Robot Operating System 2 (ROS 2). The simulated quadruped robot contains an RGB camera with a resolution of 640x480 pixels and a 2D-LiDAR with a 360° resolution of 512 rays. An environmental setup is created that varies environmental visibility and task complexity to measure the system’s adaptive performance.

Environments As illustrated in Figure 2, our system is evaluated through two experimental environments.

- **Environment A (Full Visibility):** The robot commences with all objects within the LiDAR’s sensing range, requiring the system to subsequently enrich these geometric detections with semantic attributes via the VLM.
- **Environment B (Incremental Discovery):** Significant objects are initially out of view, requiring active exploration. This environment stress-tests the new object trigger, as newly discovered entities can invalidate the current plan.

Tasks To assess the system’s adaptive capabilities and its transition from closed-world execution to open-ended discovery, we categorize the evaluation into two task types:

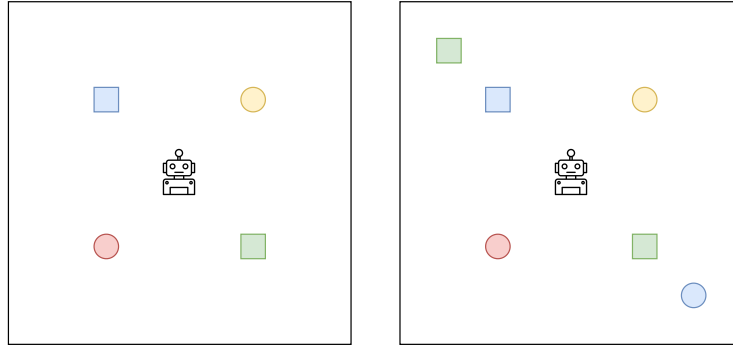


Fig. 2. Evaluation environments. Left: Environment A (Full Visibility). Right: Environment B (Incremental Discovery).

Finite Tasks These tasks incorporate explicit quantitative constraints to evaluate the monitor’s efficacy in tracking progress and mitigating state amnesia. Specifically, the robot is assigned objectives such as "Move to two squared objects", requiring consistent state maintenance and objective fulfillment.

Infinite Tasks These involve open-ended, categorical objectives, such as "Move to all squared objects", requiring the system to maintain a continuous discovery loop. Unlike finite tasks, infinite tasks lack a predefined target count. Consequently, the robot must dynamically update its execution plan whenever the perception layer identifies additional matching entities.

Operational Modalities We distinguish between goal-oriented navigation "Go to the red object" and semantic exploration "Explore until you find the red object". While both begin in unmapped environments, the former targets identified entities, whereas the latter mandates active search [9].

4.2 Benchmarking Suite

We evaluate the system using a cross-model matrix to demonstrate that the reliability of the adaptive loop is a fundamental property of the architecture rather than a specific generative model. Our selection includes frontier models optimized for code synthesis and multimodal perception, as detailed in Table 1.

4.3 Metrics

To quantify the reliability of our system, we define a set of metrics that measure its ability to maintain logical consistency while navigating an incrementally discovered environment.

Table 1. LLM and VLM configurations for the benchmarking suite.

Component	Model	Year	Parameters
Cognition	Qwen3.5-122B-A10B [23]	2026	122B
	GPT-5-nano	2025	undisclosed
	Kimi-K2.5 [22]	2026	1.1T
Perception	Qwen3-VL-8B-Instruct	2025	8B

- **End-to-End Success Rate (SR):** Evaluates the objective fulfillment of the high-level intention. This measures the system’s ability to synthesize a plan and iteratively adapt its logic as the robot’s perception of the environment evolves.
- **Syntactic Executability (SE):** Tracks how consistently the LLM generates code that adheres to operational constraints, addressing the formalization gap between fuzzy intentions and rigorous execution.
- **Re-adaptation Frequency (RF):** Quantifies the system’s reactivity by calculating the ratio of plan revisions to total trigger-induced LLM calls. The metric is defined as $RF = \frac{N_{new}}{N_{total}}$, where N_{new} is the count of structural plan updates and N_{total} is the total number of calls initiated by the adaptive triggers.

4.4 Quantitative Analysis

We evaluate the system by measuring all three metrics across both Finite and Infinite tasks, as well as across the two operational modalities. Each task is evaluated three times to account for variability in model outputs.

4.5 Baseline Robustness: Smoke Tests

Before evaluating complex navigation, we perform a series of Smoke Tests.

Table 2. Evaluation Results: Smoke Tests

#	Task Description	Model	Env. A			Env. B		
			SR	SE	RF	SR	SE	RF
1	Go to 2 -3.	Qwen3.5	1.00	1.00	0.39	1.00	1.00	0.47
		GPT-5-nano	1.00	1.00	0.58	0.67	1.00	0.75
		Kimi-K2.5	1.00	1.00	0.53	1.00	1.00	0.93
2	Move to 2 -3.	Qwen3.5	1.00	0.83	0.57	1.00	1.00	0.40
		GPT-5-nano	1.00	0.67	0.50	1.00	0.83	0.33
		Kimi-K2.5	1.00	1.00	1.00	1.00	1.00	1.00

Table 2 demonstrates the system’s resilience against adversarial prompts. Across all models and environments, near-perfect success rates and high syntactic executability confirm that formalization constraints effectively prevent malformed outputs.

4.6 Task Complexity and Model Performance

The core evaluation explores the shift from Finite Tasks to Infinite Tasks, testing the system’s ability to maintain an open-ended discovery loop as the robot uncovers its environment.

Table 3. Evaluation Results: Finite Tasks

Modality	#	Task Description	Model	Env. A			Env. B		
				SR	SE	RF	SR	SE	RF
Goal-Oriented	3	Go to a red object.	Qwen3.5	0.67	1.00	0.69	0.67	0.89	0.87
			GPT-5-nano	0.33	0.73	0.57	0.33	0.83	0.15
			Kimi-K2.5	0.67	1.00	0.74	0.00	1.00	1.00
	4	Go to two squared objects.	Qwen3.5	1.00	0.97	0.74	0.33	0.98	0.92
			GPT-5-nano	0.33	1.00	0.57	0.67	0.54	0.50
			Kimi-K2.5	1.00	1.00	0.75	1.00	1.00	0.83
	5	Go to two squared objects and tell me their colors.	Qwen3.5	0.67	0.81	0.95	1.00	1.00	0.89
			GPT-5-nano	0.33	0.66	0.96	0.67	0.56	0.68
			Kimi-K2.5	0.67	1.00	1.00	1.00	1.00	0.91
Semantic Exp.	6	Explore until you find a red object.	Qwen3.5	1.00	0.95	0.79	1.00	0.97	0.84
			GPT-5-nano	0.67	0.77	0.52	0.67	0.78	0.79
			Kimi-K2.5	1.00	1.00	0.91	1.00	1.00	0.70
	7	Explore until you find two squared objects.	Qwen3.5	0.67	1.00	0.78	1.00	1.00	0.89
			GPT-5-nano	0.33	0.94	0.55	0.67	0.63	0.65
			Kimi-K2.5	1.00	1.00	0.70	1.00	1.00	0.84
	8	Explore until you find two squared objects and tell me their colors.	Qwen3.5	1.00	0.93	0.87	1.00	0.97	0.88
			GPT-5-nano	0.00	0.55	0.49	0.00	0.68	0.81
			Kimi-K2.5	1.00	1.00	0.93	1.00	1.00	0.91

The results in Table 3 show that the proposed adaptive loop reliably preserves task progress and logical consistency in both environments. Across all configurations, the open-source models (Qwen3.5 and Kimi-K2.5) achieve higher success rates, especially in Environment B, indicating stronger robustness to dynamic observation updates and newly discovered objects. GPT-5-nano, by contrast, performs worse on tasks requiring counting and multi-step state tracking, suggesting sensitivity to the temporal state gap. Failures were concentrated in object-referenced and exploration-based tasks, pointing to systematic weaknesses in grounding, memory, and output compliance. In Tasks 3 to 5, several models

struggled sometimes to correctly map linguistic object descriptions to perceived entities. GPT-5-nano often focused on incorrect targets, hallucinated colors, or violated the response format. Kimi-K2.5 showed similar grounding failures and sometimes falsely concluded that no red objects were present under incremental discovery. Qwen3.5 was occasionally distracted by non-target objects. The results demonstrate high system reliability, with successful task completion achieved in the vast majority of cases. Residual failures are attributable not to low-level action execution, but to specific challenges in semantic grounding, constraint adherence, and the maintenance of coherent internal state representations during sequential exploration.

Table 4. Evaluation Results: Infinite Tasks

Modality	#	Task Description	Model	Env. A			Env. B		
				SR	SE	RF	SR	SE	RF
Goal-Oriented	9	Go to all cylindric objects.	Qwen3.5	0.67	0.93	0.93	0.00	1.00	0.93
			GPT-5-nano	0.33	0.87	0.70	0.00	1.00	0.57
			Kimi-K2.5	0.00	1.00	0.83	1.00	0.96	0.69
Semantic Exp.	10	Explore until you find all cylindric objects.	Qwen3.5	1.00	1.00	0.54	1.00	0.88	0.78
			GPT-5-nano	0.67	0.80	0.56	0.33	0.71	0.41
			Kimi-K2.5	0.67	1.00	0.72	0.67	1.00	0.69
	11	Explore until you find all green squared objects.	Qwen3.5	0.67	1.00	0.44	1.00	0.92	0.71
			GPT-5-nano	0.00	0.89	0.87	0.00	0.67	0.83
			Kimi-K2.5	0.67	1.00	0.73	0.67	1.00	0.77

Table 4 shows the system’s performance on open-ended tasks under continuous environment expansion. Compared with finite tasks, Success Rates are more variable, with the strongest degradation in Goal-Oriented settings, especially in Environment B, where incomplete coverage of newly discovered objects becomes a key challenge. Nevertheless, every task was solved successfully in at least 2 out of 3 cases by at least one model, showing that none was inherently infeasible or due to luck. However, strong aggregate differences remain: Qwen3.5 and Kimi-K2.5 each achieve an overall success rate of about 82%, clearly outperforming GPT-5-nano at about 46%. More specific, semantic exploration produces more consistent results overall, although GPT-5-nano still lags behind in reliability. In Tasks 9 to 11, failures reveal limitations in completeness checking, category-level reasoning, and termination control. GPT-5-nano frequently failed to cover relevant targets, stopped prematurely, or concluded that exploration was complete despite insufficient evidence. Kimi-K2.5 and Qwen3.5 showed similar but less severe issues, including partial coverage, revisiting previously explored objects, and terminating despite remaining uncertainty or unidentified objects. Overall, these results suggest that exhaustive search over a target class is challenging primarily because the models struggle to estimate environmental completeness. Beyond these qualitative nuances, the computational efficiency of these models

becomes clear when analyzing cognition latency. As shown in Figure 3, GPT-5-nano also performs worse than both open-source models when adaptive cognition duration is considered. In conclusion, the Qwen3.5 model is the most robust, with a Success Rate that is equivalent to that of the Kimi-K2.5 model. It also has the shortest inference time and the fastest adaptability, despite having the smallest parameter count. While both models utilize a Mixture-of-Experts architecture, Qwen3.5 activates significantly fewer parameters per token than Kimi-K2.5, which is responsible for its lower inference latency despite comparable success rates. It is evident that the substantial inference time per query, in conjunction with the necessity for manual evaluation of model outputs, necessitates the implementation of a maximum of three attempts per model. While this restricts the statistical robustness of the findings, it is a necessary trade-off given the computational and manual nature of the evaluation protocol.

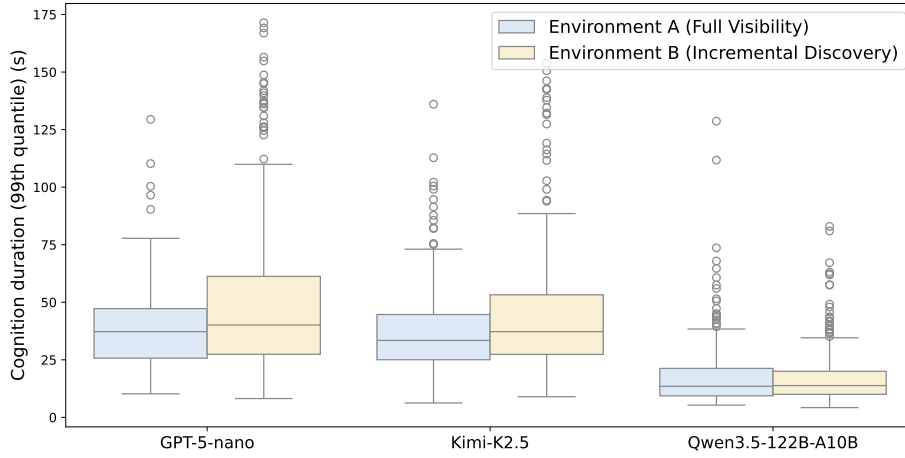


Fig. 3. Cognition durations per model and environment

5 Related Work

The engineering of Collective Adaptive Systems and the integration of generative AI into autonomous control loops represent a critical convergence of formal methods and adaptive robotics. This section reviews the foundations of methodologies for bridging the taxonomy, formalization, and temporal state gaps.

Collective Adaptive Systems, often referred to as ensembles, consist of heterogeneous components that adapt to open-ended environments through local interactions [24]. A foundational paradigm for managing these interactions is attribute-based communication, which allows components to interact based on runtime properties and roles rather than fixed identifiers [1]. In the domain of

robotics, the Protease 2.0 framework provides a parameterizable algorithmic pattern for swarm formation flight, utilizing the ATGC (Aggregation, Termination, Grouping, Calculation) pattern to enable decentralized self-organization [15]. Research into swarm robustness further utilizes statistical model checking to quantify the resilience of consensus-reaching behaviors in the presence of disruptive individuals [14].

The taxonomy gap, the inherent limitation of static command libraries in unpredictable worlds, is increasingly addressed by leveraging Vision-Language Models for open-ended scene understanding. Modern Vision-Language-Action (VLA) architectures, with VLA architectures initially pioneered by Google with the RT-2 model [27], such as the Avi model, propose reframing action generation as a problem of 3D volumetric inference over point clouds rather than 2D policy learning, providing robust spatial grounding [19]. This includes models such as 3D-VLA [25] and Any3D-VLA [10], which enable three-dimensional interaction. A further step in this direction is done by the V-jepa 2 model, which enables understanding, prediction, and planning based on a video stream [5].

The formalization gap denotes the disconnect between imprecise natural language intentions and rigorous robotic execution. AISoLA research identifies LLM-assisted software engineering as a paradigm shift where the model serves as a development expert to resolve specification ambiguities and generate implementation artifacts [6]. To ensure correctness, methodologies such as test-based refinement utilize property-based testing (e.g., ScalaCheck) to iteratively validate the semantic adherence of synthesized code [3]. Furthermore, domain-specific modeling techniques integrate structured prompt templates to constrain LLM outputs within verifiable boundaries [21].

Autonomous systems encounter a temporal state problem due to the lack of structured memory in generative controllers. Code-as-Monitor frameworks mitigate this by treating visual programming as a reactive, proactive failure-detection mechanism [26]. For collective systems, white-box validation techniques have been developed that integrate statistical model checking with process mining to graphically explain observed behaviors and identify unexpected deadlocks or synchronization errors during mission execution [8].

6 Conclusion

This work introduces a MAPE-K-inspired framework architecture that enables autonomous robots to perform tasks in unknown and unstructured environments based on high-level user intentions rather than explicit programming. It integrates a dual-AI system consisting of an LLM for code-based planning and a VLM for distilled semantic perception, which collectively address the formalization, taxonomy, and temporal state gaps that arise when using natural language as task description.

The evaluation of a quadruped robot in the context of exploration tasks has demonstrated that the architecture ensures syntactic reliability to a high degree and exhibits robust performance in the presence of finite tasks. Neverthe-

less, infinite and open-ended tasks reveal unresolved challenges in long-horizon reasoning, state awareness, and completeness checking. Specifically, while open-source frontier models exhibited superior performance in maintaining execution context, performance degradation in open-ended tasks was primarily attributed to cognitive limits inherent in exhaustive search, as opposed to invalid control commands or execution errors.

Subsequent research will address the temporal state problem with greater rigor. The intention is to move beyond simple execution pointers by implementing formal state-machine verification, where the LLM-generated plan is dynamically mapped to a state-transition system to ensure consistency across multi-step objectives. Finally, the exploration of multi-agent coordination will be conducted, with the objective of enabling the system to scale its collective semantic understanding across distributed robotic platforms operating in large, unmapped environments.

References

1. Abd Alrahman, Y., De Nicola, R., Loreti, M.: Programming the interactions of collective adaptive systems by relying on attribute-based communication. *ACM Transactions on Software Engineering and Methodology* (2017)
2. Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al.: Do as i can, not as i say: Grounding language in robotic affordances. In: *Conference on Robot Learning*. pp. 287–306. PMLR (2022)
3. Aichernig, B.K., Havelund, K.: Ai-assisted programming with test-based refinement. In: *International Conference on Bridging the Gap between AI and Reality*. pp. 385–411. Springer (2023)
4. Arcaini, P., Riccobene, E., Scandurra, P.: Modeling and analyzing mape-k feedback loops for self-adaptation. In: *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. pp. 13–23. IEEE (2015)
5. Assran, M., Bardes, A., Fan, D., Garrido, Q., Howes, R., Muckley, M., Rizvi, A., Roberts, C., Sinha, K., Zhohus, A., et al.: V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*
6. Belzner, L., Gabor, T., Wirsing, M.: Large language model assisted software engineering: prospects, challenges, and a case study. In: *International conference on bridging the gap between AI and reality*. pp. 355–374. Springer (2023)
7. Bendale, A., Boulton, T.: Towards open world recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1893–1902
8. Casaluze, R., Burattin, A., Chiaromonte, F., Lafuente, A.L., Vandin, A.: White-box validation of quantitative product lines by statistical model checking and process mining. *Journal of Systems and Software* **210**, 111983 (2024)
9. Chaplot, D.S., Gandhi, D., Gupta, A., Salakhutdinov, R.: Object-goal navigation using goal-oriented semantic exploration. In: *Proceedings of the Neural Information Processing Systems (NeurIPS)*. vol. 33, pp. 4247–4258 (2020)
10. Fan, X., Deng, S., Wu, X., Lu, Y., Li, Z., Yan, M., Zhang, Y., Zhang, Z., Wang, H., Zhao, H.: Any3d-vla: Enhancing vla robustness via diverse point clouds. *arXiv preprint arXiv:2602.00807* (2026)

11. Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G.: A survey of deep learning techniques for autonomous driving. *Journal of field robotics* **37**(3), 362–386 (2020)
12. Index, T.: Tiobe index (2023)
13. Jiang, Y.q., Zhang, S.q., Khandelwal, P., Stone, P.: Task planning in robotics: an empirical comparison of pddl-and asp-based systems. *Frontiers of Information Technology & Electronic Engineering* **20**(3), 363–373 (2019)
14. Klein, J., d’Onofrio, A., Petrov, T.: Exploring consensus robustness in swarms with disruptive individuals. In: *ISoLA 2024 Proceedings*. Springer (2024)
15. Kosak, O., Kastenmüller, P., Wanninger, C., Reif, W.: An approach for extended swarm formation flight with drones: Protease 2.0. In: *International Symposium on Leveraging Applications of Formal Methods*. pp. 263–280. Springer (2024)
16. Kroemer, O., Niekum, S., Konidaris, G.: A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research* **22**(30), 1–82 (2021)
17. Moon, H., Seo, J., Lee, S., Park, C., Lim, H.: Find the intention of instruction: Comprehensive evaluation of instruction understanding for large language models. *arXiv preprint arXiv:2412.19450* (2024)
18. Qin, Y., Song, K., Hu, Y., Yao, W., Cho, S., Wang, X., Wu, X., Liu, F., Liu, P., Yu, D.: Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601* (2024)
19. Song, H., Le, L.: Avi: A 3d vision-language action model architecture generating action from volumetric inference. In: *NeurIPS 2025 Workshop on Embodied World Models for Decision Making*
20. Sridharan, M., Gelfond, M., Zhang, S., Wyatt, J.: Reba: A refinement-based architecture for knowledge representation and reasoning in robotics. *Journal of Artificial Intelligence Research* **65**, 87–180 (2019)
21. Steffen, B. (ed.): Bridging the Gap Between AI and Reality - Second International Conference, AISoLA 2024, Crete, Greece, October 30 - November 3, 2024, Selected Papers, Lecture Notes in Computer Science, vol. 16032. Springer (2026). <https://doi.org/10.1007/978-3-032-01377-4>, <https://doi.org/10.1007/978-3-032-01377-4>
22. Team, K.: Kimi k2.5: Visual agentic intelligence (2026), <https://arxiv.org/abs/2602.02276>
23. Team, Q.: Qwen3.5-omni technical report (2026), <https://arxiv.org/abs/2604.15804>
24. Wirsing, M., De Nicola, R., Jähnichen, S., Tribastone, M.: Rigorous engineering of collective adaptive systems – 2nd special section. *International Journal on Software Tools for Technology Transfer* **25**(2), 1–8 (2023)
25. Zhen, H., Qiu, X., Chen, P., Yang, J., Yan, X., Du, Y., Hong, Y., Gan, C.: 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*
26. Zhou, E., Su, Q., Chi, C., Zhang, Z., Wang, Z., Huang, T., Sheng, L., Wang, H.: Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 6919–6929 (2025)
27. Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohllhart, P., Welker, S., Wahid, A., et al.: Rt-2: Vision-language-action models transfer web knowledge to robotic control. In: *Conference on Robot Learning*. pp. 2165–2183. PMLR (2023)