

Distributed AI Agents for Cognitive Underwater Robot Autonomy

Markus Buchholz, Ignacio Carlucho, Michele Grimaldi, and Yvan R. Petillot

Abstract—Achieving robust cognitive autonomy in robots navigating complex, unpredictable environments remains a fundamental challenge in robotics. This paper presents Underwater Robot Self-Organizing Autonomy (UROSA), a groundbreaking architecture leveraging distributed Large Language Model AI agents integrated within the Robot Operating System 2 (ROS 2) framework to enable advanced cognitive capabilities in Autonomous Underwater Vehicles. UROSA decentralises cognition into specialised AI agents responsible for multimodal perception, adaptive reasoning, dynamic mission planning, and real-time decision-making. Central innovations include flexible agents dynamically adapting their roles, retrieval-augmented generation utilising vector databases for efficient knowledge management, reinforcement learning-driven behavioural optimisation, and autonomous on-the-fly ROS 2 node generation for runtime functional extensibility. Extensive empirical validation demonstrates UROSA’s promising adaptability and reliability through realistic underwater missions in simulation and real-world deployments, showing significant advantages over traditional rule-based architectures in handling unforeseen scenarios, environmental uncertainties, and novel mission objectives. This work not only advances underwater autonomy but also establishes a scalable, safe, and versatile cognitive robotics framework capable of generalising to a diverse array of real-world applications.¹

I. INTRODUCTION

The pursuit of truly autonomous robots capable of effectively navigating and interacting within complex, unstructured environments remains a grand challenge in the field of robotics [1]. For decades, the dominant paradigm has been rooted in traditional robotic systems, which rely on modular architectures and meticulously pre-defined, rule-based algorithms [2]. While these systems have demonstrated notable success in controlled and predictable settings such as factory automation, they increasingly reveal their limitations when confronted with the inherent dynamism and uncertainty of real-world scenarios [3].

Legacy approaches often struggle with novel or unforeseen situations, demanding extensive manual reprogramming for even minor environmental changes and fundamentally lacking the adaptability needed for open-ended tasks [4]. This inherent inflexibility prevents them from truly operating in a *self-playing* manner, where the system autonomously adapts and solves problems without continuous human intervention.

Within this evolving landscape, the rapid advancement of Large Language Models (LLMs) and Vision Language Models (VLMs) offers a compelling pathway toward more flexible, adaptive, and robust robot autonomy. Unlike conventional

software, which executes a rigid set of programmed instructions, AI agents are designed to understand complex information, reason about it, and generate a sequence of actions to perform specific tasks [5], [6]. This approach fundamentally shifts the development paradigm from imperative programming, where every step of a task is explicitly coded (e.g., *move forward 1 meter, turn left 90 degrees*), to declarative goal setting, allowing the agent to autonomously determine how to fulfill its mission (e.g., *inspect the valve*). By leveraging vast pre-trained knowledge repositories, AI agents exhibit emergent reasoning capabilities that can adapt to new tasks and conditions without requiring exhaustive manual reprogramming [7], [8]. However, the integration of these agents in robotic applications introduces new challenges, such as ensuring robustness and verifiability of the decision made and protecting against possible *hallucinations* [9], [10]. The increasing capabilities of advanced AI systems also bring to the forefront critical considerations related to AI safety and verification, encompassing complex issues such as alignment of the mental models of the operators and the AI system [11]–[13].

UROSA is inspired by ROSA [11], which is used as a parser from high level natural language to generate actions through ROS messages. Unlike ROSA, our work integrates agentic AI at multiple levels of the autonomy architecture, able to make decisions as well as communicate through ROS. This is a novel and highly capable autonomy framework that fundamentally *replaces the traditional paradigm of a human-governed main program with a distributed network of specialised and dynamically adaptable AI agents*. This distributed architecture represents a radical shift towards true autonomy, where once a mission is initiated with high-level commands, the system operates with minimal human interaction and oversight. The distributed AI agents communicate and solve problems together, without needing human intervention or the development of coding or extra programmatic loops. This allows for emergent system-level intelligence, where human interaction primarily involves providing high-level commands through natural language. Each agent is responsible for a specific aspect of the robot’s operational workflow, ranging from multimodal perception (e.g., vision, depth, sonar) to high-level strategic planning. Crucially, the AI agents within UROSA are not merely generative; they are designed as **agentic AI entities**, capable of autonomous decision-making and action-taking within their environment without requiring continuous human intervention in the loop [14], [15].

The realisation of this distributed cognitive architecture is founded upon the following key innovations that form the core of our UROSA framework:

M. Buchholz, I. Carlucho, M. Grimaldi, and Y.R. Petillot are with the School of Engineering & Physical Sciences, Heriot-Watt University, Edinburgh, UK. e-mail: m.buchholz@hw.ac.uk.

¹UROSA project webpage: <https://markusbuchholz.github.io/urosa.html>

- **Decoupled Reasoning and Environmental Adaptability:** The framework achieves architectural flexibility by replacing traditional code-based logic with pretrained AI agents that handle system functions. Domain experts, for example a Remotely Operated Vehicle (ROV) pilot, can build systems using these existing agents without extensive reengineering. The system is adaptable in near-real time to a large variety of changes in the environment and internal states, and can access a much larger set of data through its ability to understand descriptions in natural language accessible from a variety of sources, e.g., live meteorological ocean data, vehicle design.
- **Behaviour adaptation and lifelong learning:** The system is also able to learn and adapt in real-time to improve existing agentic behaviour or generate new ones, based on prompts from operators or other agents. It can also improve over time by using previous experiences and operator feedback. This flexibility is supported by a Vector Database (VDB) that stores and retrieves past experiences, observational data, simulator outcomes and external knowledge. This comprehensive data forms the foundation for Retrieval-Augmented Generation (RAG), facilitating context-driven and informed decision-making.
- **Autonomous On-the-Fly Function Extension:** The system can generate code on the fly to deal with unforeseen circumstances to extend its functionality dynamically based on real-time requirements identified autonomously by the Planning Agent. This means that if a new functional component is needed for the mission, the Planning Agent can request its generation, testing, and integration without human code intervention. These capabilities can be further refined through online, agent-driven instructional tuning, where agents learn optimal response strategies through iterative, interactive feedback. The new code can be validated in simulation before being deployed on the real platform. This enables unprecedented adaptability in response to evolving mission demands and environmental contingencies.
- **Dynamic, Predictive System Diagnostics:** The system is able to perform diagnostics of its internal state and environmental conditions, without the requirement of a predefined static fault tree or a fixed set of unit tests.
- **Inherent Safety and Control:** UROSA explicitly addresses aspects of AI safety and control. Through the introduction of several robust ROS 2 mechanisms and architectural constraints, such as a dedicated safety parser and a layered design, we introduce fine-grained control over the agents' outputs and behaviours. This framework aims to mitigate hallucinations and increase the likelihood of actions aligning with predefined safety protocols, thereby contributing to the critical goal of aligning AI with human values in autonomous operations.

While UROSA leverages contemporary LLMs and VLMs, its foundational architecture is inherently model-agnostic and designed with a forward-looking perspective. Recognising the rapid advancements in artificial intelligence its modular design allows for the seamless integration of future, more advanced

AI paradigms as they emerge. We aim to propose a general concept for distributed cognitive autonomy that can evolve and benefit from future breakthroughs in AI, ensuring its continued relevance and adaptability.

We validate each of the key innovations previously outlined through a series of demanding use-case scenarios. These empirical tests provide concrete evidence of the system's performance in areas such as real-time adaptation, autonomous code generation, continuous learning, and advanced diagnostics. To further illustrate these capabilities, this paper is accompanied by a supplementary video² demonstrating our key experiments in both simulation and real-world deployments. Finally, we analyse these results to address the critical challenges of AI safety and verifiability, and discuss the future trajectory of distributed cognitive architectures like UROSA in an era of rapidly advancing artificial intelligence.

II. LITERATURE OVERVIEW: THE ASCENT OF COGNITIVE AUTONOMY IN ROBOTICS

The pursuit of cognitive autonomy has historically navigated a path between symbolic reasoning and reactive, embodied intelligence. Early endeavours, influenced by symbolic AI, sought to replicate cognition through formal logic and planning [2], [16], [17], but these top-down systems often proved unstable when faced with the uncertainty and dynamism of the real world [3], [18], [19]. In response, a bottom-up approach emerged with behaviour-based robotics and the principle of embodied, situated cognition, which emphasised that intelligence arises from the direct sensorimotor interaction between an agent and its environment [20]–[23]. While architectures like subsumption produced remarkably robust reactive behaviours [4], they lacked the capacity for abstract reasoning and complex planning. Cognitive architectures such as SOAR [7] and ACT-R [8], [24] attempted to bridge this gap by integrating symbolic processing with more plausible cognitive mechanisms, yet often struggled with the knowledge acquisition bottleneck and extensive hand-engineering required for novel domains [25].

The recent advent of LLMs has been a paradigm shift, offering powerful new avenues for high-level robotic cognition [26]. Trained on vast datasets, models like GPT-4 have demonstrated remarkable emergent capabilities in reasoning and natural language understanding [5], [6], [27], [28], making them potent tools for robotics [29]. Research has rapidly demonstrated their efficacy in translating high-level natural language commands into actionable plans, as seen in works like SayCan [30], [31]. This has been extended by grounding language in rich perceptual data using VLMs [32]–[36], generating executable robot code from text [37], [38], and leveraging interactive reasoning paradigms such as ReAct [39] for complex, embodied problem-solving.

However, integrating LLMs effectively and safely into robotic systems introduces a new set of research challenges, moving the frontier from single-model planning to robust, multi-agent architectures. The concept of a single, monolithic *robot brain* gives way to distributed cognitive architectures, which promise greater modularity, fault tolerance, and emergent

²The video is available at: <https://markusbuchholz.github.io/urosa.html>

intelligence from the coordination of multiple specialised agents [40]–[43]. Realising this vision of true cognitive autonomy [44]–[46] requires overcoming significant hurdles. These include ensuring robustness against model *hallucinations* [9], [10], [47], achieving real-time performance on resource-constrained hardware [48], and addressing the critical challenges of AI safety, verification, and ethical alignment in safety-critical applications [49]–[52].

This paper introduces UROSA, a novel distributed AI agent architecture designed to directly address this new frontier. Our work moves beyond using a single LLM for planning and instead proposes a collaborative ecosystem of specialised agentic nodes integrated deeply within the ROS 2 framework. By decentralising cognition, UROSA enhances robustness and scalability. It explicitly tackles the challenges of reliability and safety through mechanisms such as RAG for factual grounding, per-agent safety parsers to validate outputs, and dynamic, on-the-fly function generation to adapt to unforeseen circumstances. Through the design and extensive empirical validation of this framework, we demonstrate a concrete and scalable pathway toward more capable, adaptable, and reliable cognitive autonomy in real-world robotic systems.

III. ARCHITECTURE

The UROSA framework fundamentally reimagines robotic autonomy by replacing a single, monolithic control program with a distributed cognitive architecture, as depicted in the overview in Fig. 1. This design is rooted in the principles of embodied and situated cognition [21]–[23], where intelligence emerges from the dynamic interaction between multiple specialised agents and their environment. At its core, UROSA decentralizes high-level reasoning across a network of AI agents, each functioning as an intelligent, collaborative unit. These agents leverage the robust communication backbone of ROS 2 to coordinate their actions and share knowledge. This hybrid architecture follows a clear separation: deterministic, low-level controllers handle time-critical tasks like stabilization, while the distributed AI agents perform the high-level cognitive functions of reasoning, planning, and adapting to novel situations.

The fundamental building block of this architecture is the *Agentic ROS 2 Node*. As detailed at the top right of Fig. 1, this is not a standard ROS 2 node but a composite entity. It fuses the high-level reasoning of an *AI Agent* with the robust communication of its ROS 2 Node Implementation. Each of these nodes encapsulates the core **AI Reasoner**, a critical **Safety Parser** to validate all outputs, and standard **Publishers/Subscribers** for system-wide communication. This modular design is a core innovation of UROSA, allowing specialised intelligence to be embedded directly within the robotics framework.

A. The Set of Cognitive Intelligent Agents

The cognitive core of UROSA is not a single *brain* but a distributed team of agents with distinct roles, all communicating over ROS 2.

a) *The Commander AI Agent*: This agent serves as the central cognitive orchestrator, akin to a vessel’s captain. Upon receiving a high-level mission description, it utilises techniques inspired by *Chain-of-Thought* prompting to decompose the goal into a coherent sequence of sub-tasks for the specialist agents [53]. Its primary role is to maintain cognitive consistency by employing a reflection and revision cycle, similar to the *self-correction* processes described in state-of-the-art agent architectures [54].

b) *The Autonomous Crew (Specialist Agents)*: A suite of specialist agents, analogous to an expert crew, handles specific functional domains. The agents shown in Fig. 1 are key examples, including the **Perception & Scene Reasoning Agent**, the **Motion Planning Agent**, the **Autonomous Code Synthesis Agent**, the **Predictive Diagnostics Agent**, the **Capability Assessment Agent**, and the **Digital Twin Curator Agent**. This set is not fixed; the modularity of the framework allows for the seamless integration of new agents as mission complexity demands.

c) *Shared Resources (ROS 2 & VDB)*: All agents are connected by the ROS 2 framework, which serves as the distributed **communication backbone** for the entire system. Furthermore, the VDB serves as a distributed, long-term memory accessible by all agents, enabling them to learn and improve their performance over time through RAG.

B. Operational Flow: Digital Twin and Real-World Interaction

A typical mission unfolds in two interconnected loops:

1. Cognitive Control Loop: The *Commander Agent* receives a mission and directs the specialist agents to generate plans and actions. For instance, the *Motion Planning Agent* generates a trajectory, which is passed to the low-level *Controls & Interfaces* to be executed by the robot in the *Real World*.

2. Digital Twin Loop: Concurrently, the robot’s *Sensors* provide real-time *Meta data* to the *Digital Twin Curator Agent*. This agent reasons about the real-world state and sends *Fidelity Injections*-updates and corrections-to the *Simulator*. The simulator’s *Virtual State*, now synchronised with reality, is used to create a high-fidelity *Digital Twin* for predictive analysis and proactive planning.

IV. MECHANISMS

This section provides a detailed description of UROSA’s main mechanisms, which were initially introduced in Section I. Section V will then present different use cases that demonstrate the performance of each mechanism.

A. Decoupled Reasoning and Environmental Adaptability

A primary innovation of UROSA is its ability to decouple high-level mission goals from low-level code implementation. This is achieved not just by distributing agents, but by fundamentally replacing traditional programmatic logic with a structured, pre-configured reasoning process embedded within each AI agent.

The other innovation is the adaptability that these agents allow. The main engine behind UROSA’s adaptability is the

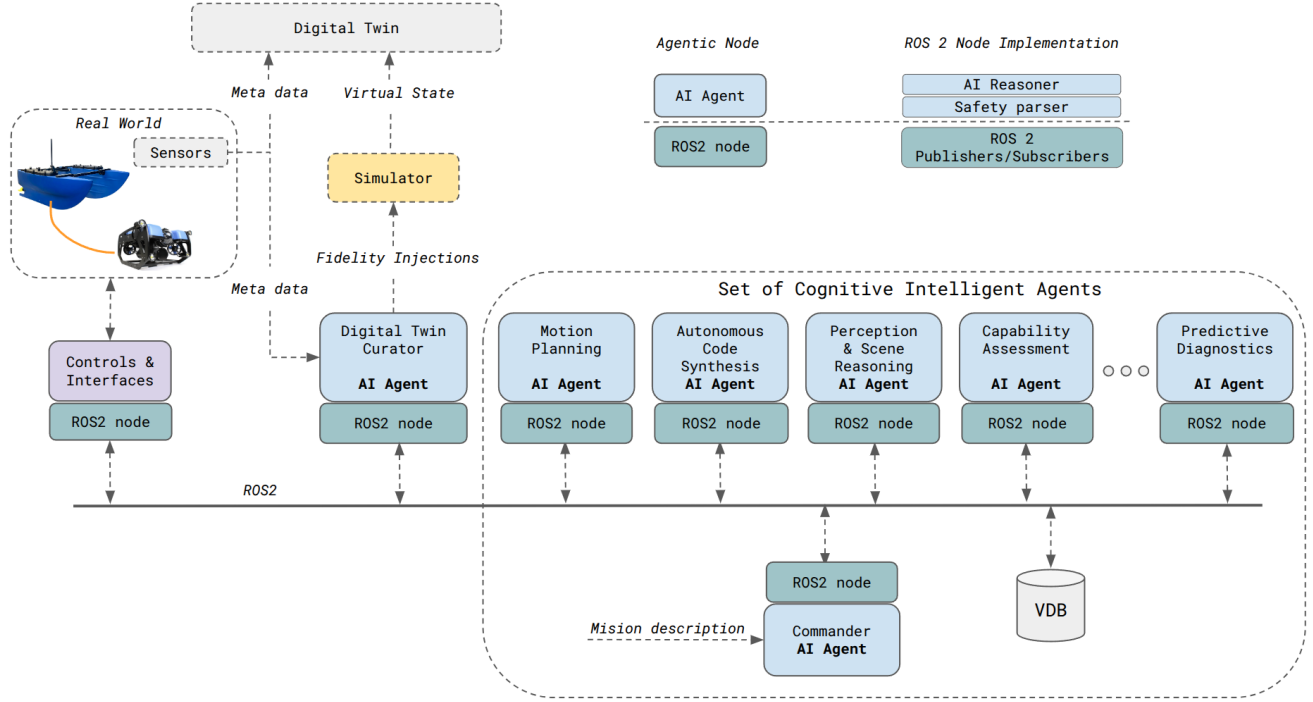


Fig. 1: The UROSA Cognitive Architecture. The system is comprised of three main parts: (left) the real-world vehicle and its low-level controls; (top) the Digital Twin environment, which is continuously updated by a dedicated curator agent; and (center) the cognitive core of the framework, a distributed set of intelligent AI agents. The **Commander AI Agent** receives the high-level mission description and acts as a central orchestrator, ensuring cognitive consistency. It directs a crew of **Specialist Agents**, each responsible for a specific domain like planning, perception, or diagnostics. All agents communicate over the ROS 2 bus and access a shared, distributed **VDB** for long-term memory and experiential learning. The **Digital Twin Curator Agent** observes real-world sensor data *Meta data* and injects updates *Fidelity Injections* into the *Simulator*, whose *Virtual State* is fused with real-world data to form the comprehensive Digital Twin. The inset (top right) details the composition of each **Agentic Node**, which fuses the high-level **AI Agent** with its **ROS 2 Node Implementation**, encapsulating the **AI Reasoner**, a **Safety Parser**, and communication interfaces.

pre-deployment *tuning* of each agent for its specific role. This is achieved not by changing model weights, but by engineering a detailed set of instructions within the `SYSTEM` prompt of its *Modelfile*. This prompt acts as the agent’s core cognitive and behavioural model, containing:

- **A Core Directive:** Defining its primary function and role within the multi-agent system (e.g., *You are a motion planner* or *You are a follower agent that must coordinate with a master agent*).
- **Domain Knowledge and Physical Models:** Textual descriptions of vehicle dynamics, environmental forces, and physical constraints like tether properties.
- **Reasoning Guidelines:** Instructions on how to approach a problem, covering both individual and cooperative logic (e.g., *You must compute a collision-free path that yields to the master agent’s trajectory*).
- **A Strict Output Format:** A rigid template for its response, as detailed in our safety strategy (Section V-D)

In operation, the agent’s ROS 2 node subscribes to relevant topics and passes this multi-modal information to the tuned AI core. The AI agent then executes its embedded reasoning process to formulate a solution. This entire *thinking process*, guided by the initial `SYSTEM` prompt, replaces thousands

of lines of explicit code, allowing for unprecedented flexibility.

B. Behaviour adaptation and lifelong learning

UROSA is designed for continuous learning, where agents adapt their behaviour based on new information and past experiences. We enable this capability through two distinct, complementary mechanisms: (1) learning using a VDB with RAG for knowledge-grounding, and (2) online policy refinement using a novel *Teacher-Student* model for behavioural shaping.

The primary mechanism for grounding and learning from past data is the VDB. At the start of a mission, a **Flexible AI Agent** is instantiated with a specific behaviour defined in its `SYSTEM` prompt. This initial prompt instructs the agent on its core task and, crucially, how to utilise information from the VDB. During operation, the VDB is dynamically updated with valuable, curated data. In each operational cycle, the agent queries the VDB to retrieve contextually relevant past experiences. This RAG process grounds the agent’s reasoning, dramatically improving decision quality and accelerating performance by leveraging a growing repository of relevant experience.

To facilitate more direct online behavioural refinement, we introduce a novel **Teacher-Student Instructional Tuning** mechanism. This approach enables autonomous adaptation of a Student agent's core policy through direct interaction with a Teacher agent. The process unfolds over episodes: the Student performs an action (e.g., generates a description). The Teacher agent analyses the response and its corrective action is to generate a new, modified `SYSTEM` prompt for the Student. This prompt acts as a rich, instructive policy guidance signal. This new prompt is then used to create a new, re-tuned instance of the Student agent for the next episode, creating a powerful meta-learning loop where the Teacher actively shapes the Student's core reasoning policy from one episode to the next.

C. Autonomous On-the-Fly Function Extensibility

A key innovation of UROSA is its ability to autonomously generate, test, and integrate new software components at runtime, extending its own functionality without direct human intervention. This capacity for self-extension allows the system to adapt its software architecture to unforeseen challenges or mission requirements, effectively enabling runtime *self-repair*.

This mechanism is enabled by the **Autonomous Code Synthesis AI Agent**, operating under the direction of the *Commander AI Agent*. The process begins when the *Commander* autonomously identifies a functional gap in the system, for instance, determining that a specific data filter or a novel planning algorithm is required to handle the current situation. It then formulates a set of high-level requirements and sends them as a natural language request to the Code Synthesis agent.

It is crucial to note that the AI's role is strictly that of a powerful code synthesiser; it does not perform the requested task itself. Upon receiving the requirements, the agent: **(1) Synthesises Code:** Generates the source code for a complete ROS 2 node in a suitable language like Python; **(2) Generates Tests:** Simultaneously creates a suite of unit tests to validate the new code's functional correctness and safety; **(3) Saves and Deploys:** If the automated tests pass, the generated source code is saved as a new executable file. This new node is then seamlessly launched and integrated into the live ROS 2 computation graph. This entire workflow (from the identification of need to the deployment of a validated new capability) is performed on-the-fly.

D. Dynamic, Predictive System Diagnostics

UROSA features an advanced diagnostics capability that moves beyond static fault trees, using an AI agent to reason about the system's health based on live data. This allows for the detection of complex, emergent failures that are not defined by simple error codes.

This capability is enabled by a dedicated **Diagnostic AI Agent** whose behaviour is defined by a highly structured, task-specific reasoning process embedded within its core `SYSTEM` prompt. This transforms the generic LLM into a specialised diagnostic expert. The prompt instructs the agent to: **(1)** Perform a multi-step time-series analysis over a sliding window of 10 consecutive JSON status messages from the

vehicle. **(2)** Utilise an embedded domain-specific physical model-the vehicle's thruster allocation matrix-to determine the expected thruster behaviour for any commanded movement. **(3)** Compare the expected PWM values against the observed data for each thruster in the time window, using defined logic to classify distinct fault types (e.g., *dead* or *out-of-range*). **(4)** Aggregate these findings to identify drifts, asymmetries, or other anomalies. **(5)** Report its findings in a rigid, three-line format: a) Issue Identification, c) Status Summary, and b) Suggested Action.

This entire diagnostic procedure, governed by the initial prompt, allows the agent to use its holistic understanding of the vehicle's expected behaviour to diagnose issues without relying on a predefined list of faults.

E. Inherent Safety and Control Mechanisms

Finally, UROSA explicitly addresses AI safety and control through its architecture, aiming to ensure that the agents' actions are verifiable and aligned with human intent. This is achieved through a multi-layered safety strategy, implemented from the agent's creation to its final output.

Safety in UROSA is not a single component but an integrated, three-tiered approach:

1. Proactive Behavioural Scaffolding: The first and most critical layer of control is established during each agent's creation via its *Modelfile*. This involves embedding a complete, task-specific reasoning process and strict output format directly into the agent's core `SYSTEM` prompt. By providing detailed instructions, domain knowledge, physical models, and reasoning guidelines, we transform the generic LLM into a highly specialised and predictable, task-oriented engine. The detailed implementation of this principle for the *Diagnostic Agent*, as described in Section IV-D, is a prime example of this technique. This same scaffolding approach is applied to all agents in the UROSA framework to ensure reliable and verifiable behaviour.

2. Dynamic Contextual Grounding: The second layer is provided by the heavy reliance on RAG, orchestrated by the *Brain Agent*. While the `SYSTEM` prompt defines *how* an agent should reason, RAG ensures that *what* it reasons about is grounded in a verified knowledge base and real-world data from the VDB. This provides the necessary context to make its structured output factually relevant to the current mission, further mitigating the risk of ungrounded decisions.

3. Reactive Output Validation: As a final, reactive backstop, every agentic node contains a **Safety Parser**. This component validates all LLM outputs before they are published as ROS 2 messages. It acts as a critical final checkpoint, checking for correct syntax and ensuring the command adheres to predefined operational safety rules (e.g., maximum speed or depth). Any command that fails this validation is blocked.

Furthermore, the distributed architecture itself provides inherent fault tolerance, as the failure of one specialist agent does not necessarily compromise the entire system.

V. RESULTS

To substantiate the claims made in this paper, we conducted a comprehensive evaluation strategy designed to validate each

TABLE I: Dynamic Avoidance: AI Agent vs. Enhanced A*

Mission ID	Error Delta (AI - A*) (m)	AI Agent Success (%)
1	2.56	80%
2	3.78	80%
3	4.23	80%
4	6.37	60%
5	4.68	60%

of UROSA’s core innovations. The experiments were both in simulation, using in a high-fidelity simulation environment, and in real-time robotic platforms. We utilise an Autonomous Surface Vehicle (ASV) and an Autonomous Underwater Vehicle (AUV) [55], in a water tank (3.5 m x 3.0 m x 2.5 m) available in our laboratories. The following subsections are structured to address each innovation introduced in Section I and described in detail in Section IV, presenting empirical validation through targeted use-case scenarios.

A. Decoupled Reasoning and Environmental Adaptability

We validated this through several complex scenarios, covering both multi-agent coordination and single-robot manipulation, with quantitative results presented for each to demonstrate performance and reliability.

1) Constrained Multi-Robot Coordination With Obstacles:

In this simulation, a tethered ASV-AUV system (with a 10 m tether) was tasked with achieving a position goal while avoiding an obstacle, as shown in Fig. 2. This required the AI agent to reason about the complex system dynamics, including the tether’s catenary shape, to maintain a valid formation throughout the maneuver. To benchmark UROSA’s performance, we compared it against a traditional motion planner based on an enhanced A* algorithm [56], [57]. A critical distinction in this evaluation is the source of obstacle information: the traditional A* planner was provided with the ground-truth position of the obstacle, whereas the UROSA framework had to rely on a *Perception & Scene Reasoning Agent* to detect and track the obstacle from a live camera feed.

The validation was performed across 5 distinct mission configurations, each featuring a unique map with different start positions, goal locations, and obstacle layouts to test the system’s adaptability. For each of these configurations, we conducted 5 trials to account for any stochasticity in the perception and planning process. The key comparative metrics—the additional positioning error introduced by the AI agent compared to the A* baseline, and the agent’s task success rate—are presented in Table I. The results show an expected trade-off: the AI agent’s positioning error is higher because it operates on a live perception feed, unlike the A* planner, which used perfect information. Nevertheless, achieving a success rate of up to 80% on this difficult perception-in-the-loop task is a significant result, demonstrating that the framework can close the loop from raw perception to complex multi-robot coordination using only emergent reasoning.

2) *Map-Based Path Planning for Multi-Robot Systems:* To test the ability to interpret abstract data, we tasked the AI system with planning a path for the tethered ASV-AUV system using only a 2D map image and a text-based goal (Fig. 2). We again compared the AI-driven approach to the enhanced A*

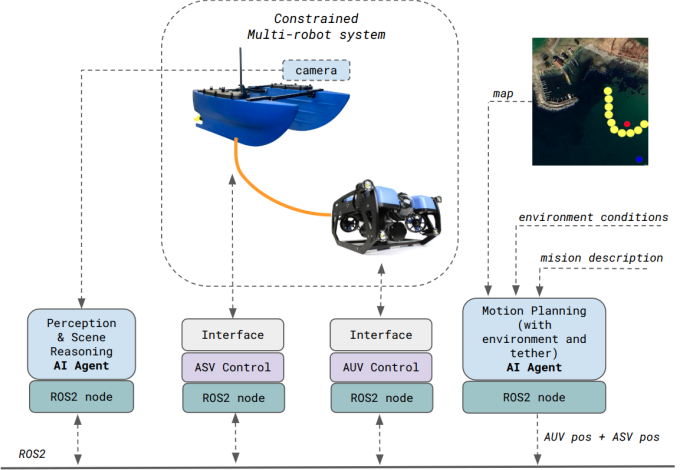


Fig. 2: Illustration of UROSA’s flexible, multi-modal planning architecture. The system’s AI agents act as a cognitive hub, integrating pre-defined mission data or dynamic, real-time perceptions to enable seamless switching between goal-driven and perception-driven behaviors. In the map-based planning scenario shown, the agent must generate a collision-free path from the **blue circle** (start position) to the **red circle** (goal position), treating the **yellow circles** as obstacles.

TABLE II: Map-Based Planning: AI Agent vs. Enhanced A*

Map ID	Error Delta (AI - A*) (m)	AI Agent Success (%)
1	3.34	80%
2	2.98	80%
3	7.81	60%
4	8.13	60%
5	6.26	40%

planner [56], [57]. The averaged results over 5 trials for each map layout are compared in Table II.

The results in Table II demonstrate the trade-offs of an end-to-end reasoning system. The higher positioning error and variable success rates are an expected consequence of the task’s complexity; the AI agent must first interpret the raw map image, identify obstacles, and then plan a path, a process with more inherent uncertainty than the A* planner’s, which operates on pre-defined maps. The key achievement is that the system can succeed in this complex task at all, reaching success rates of up to 80% on. This is a significant leap in autonomy because the UROSA framework reasons directly from raw images and text to create a plan, bypassing the time-consuming manual setup required by traditional planners.

3) *Flexible Motion Planning for a UVMS:* We further validated this principle on an Underwater Vehicle Manipulator System (UVMS) [58], [59], focusing on autonomous motion planning driven by textual commands (Fig. 4). The experiment, conducted in a real-world tank, was designed to test the *Commander AI Agent*’s ability to interpret nuanced natural language instructions. To simplify the test’s scope and focus purely on command interpretation and planning, the coordinates for key locations like *goal 1* and *goal 2* were predefined within the agent’s SYSTEM prompt.

To provide a quantitative benchmark, we compared its

TABLE III: Manipulator Planning & Interpretation Performance

Metric	AI Agent	Naive A*+PRM
Planning Time (s)	1.3	0.05
Interpretation Success (%)	90%	N/A
Planning Success (%)	100%	100%

performance against a baseline traditional planner using a naive A* search on a Probabilistic Roadmap (PRM) [60]. To test the reliability of the AI’s language understanding, we conducted 10 trials using prompts with varied phrasing and intent. These included explicit instructions for avoidance (e.g., *Go to goal 1 and check the obstacles on the way*) as well as more ambiguous phrasing (e.g., *Inspect goal 2 and check for any boxes on your way*). Crucially, the test also included direct commands that omitted any mention of obstacle avoidance, such as *Inspect goal 2*. In these cases, to test literal command adherence, the *Commander AI Agent* intentionally disregarded perceptual data about obstacles and commanded a direct path, leading to a planned collision. The averaged results of these trials are summarized in Table III.

The results highlight the unique capabilities of the UROSA approach. The system achieved a 90% *Interpretation Success* rate, correctly understanding the user’s intent in 9 out of the 10 trials. Crucially, for every prompt that was correctly understood, the subsequent *Planning Success* rate for generating a valid trajectory was 100%. This clearly distinguishes between the agent’s highly robust (but not infallible) natural language understanding and its near-perfect planning capability once the goal is known—a distinction the traditional method (N/A) cannot address. This showcases the framework’s power in translating high-level, human-like instructions directly into complex robot actions.

B. Behaviour Adaptation and Lifelong Learning

We evaluated these adaptive learning capabilities through targeted experiments demonstrating both experiential learning via RAG and online policy refinement.

1) *Visual Positioning with Experiential Learning*: To validate how the framework’s use of experience improves resilience against external disturbances, we conducted simulation tests focused on visual positioning. In this scenario, an AUV was tasked with maintaining a stable position and orientation relative to a static feature on the seabed (a pipeline segment), as shown in the simulated views in Fig. 5. This capability is enabled by a **Flexible AI Agent** that uses a VDB as a short-term visual-temporal memory. This allows the agent to reason not just about its current state, but about the dynamics of recent events.

To isolate the impact of this experience buffer, we benchmarked the agent’s performance in two conditions: (1) with VDB access disabled, and (2) with full VDB access. The test involved letting the AUV follow a pipeline and then applying a simulated external force to induce a specific lateral deviation. To evaluate the robustness of the agent’s response, we tested three magnitudes of absolute deviation (1.0 m, 1.5 m, and 2.5 m), applying the force to induce errors to both the left and the right. We then measured the *Recovery Time*—the time taken

TABLE IV: Disturbance Recovery with Experiential Learning

Deviation	Recovery Time (w/o VDB, s)	Recovery Time (w/ VDB, s)
1.0 m	5.9	2.6
1.5 m	9.0	3.2
2.5 m	12.6	4.4

TABLE V: Averaged Online Policy Refinement via Teacher-Student Loop

Episode	Avg. Response Length (Words)	Avg. Info. Relevance (%)
1	45	5%
3	18	65%
6	5	100%

for the agent to guide the AUV back to the pipe centerline. Each of these six conditions was run 3 times, and the results, averaged across both left and right directions for each deviation magnitude, are presented in Table IV.

The results demonstrate the significant value of experiential learning. The agent with VDB access recovered much faster because it leveraged the visual history to infer the disturbance dynamics and issue a proactive correction command. In contrast, the baseline agent could only react to its instantaneous error, proving less effective. The key hardware-independent finding is that the VDB provides the temporal context required for more resilient control.

2) *Online Behavioural Tuning via Teacher-Student Interaction*: To demonstrate more direct online behavioural adaptation, we validated our novel **Teacher-Student Instructional Tuning mechanism**. In this paradigm, a *Teacher* agent guides a *Student* agent’s policy by generating a new, more restrictive *SYSTEM* prompt as a form of instructive feedback, creating a powerful meta-learning loop. We conducted a series of real-world experiments in a cluttered tank environment (Fig. 4) where the *Commander AI Agent* acted as the *Teacher* and the *Perception & Scene Reasoning AI Agent* was the *Student*.

The *Teacher*’s objective was to guide the *Student*’s policy from providing verbose descriptions of the whole scene to reporting *only* the presence and location of a specific target. We ran a total of 20 trials, comprising 5 trials for each of four distinct target types: red balls, pink buoys, a fishing net, and other submerged obstacles. At each episode in a trial, the *Student* generated a description, and the *Teacher* provided corrective feedback by generating a new *SYSTEM* prompt to re-tune the *Student*’s behaviour for the next episode. The learning process was quantified over three episodes, and Table V shows the averaged performance across all 20 trials.

The results demonstrate a consistent and rapid convergence on the desired behaviour across all target types. On average, the *Student*’s policy quickly shifted from general scene description to specific, targeted reporting, confirming the mechanism’s ability to perform targeted, online behavioural shaping for a variety of perceptual goals using structured, linguistic guidance.

3) *Validation*: We evaluated this mechanism’s capacity for runtime software adaptation and self-repair using the controlled test setup shown in Fig. 6. For this evaluation, we simulated an autonomous request from the *Commander AI Agent* by manually publishing a ROS 2 topic containing the high-level

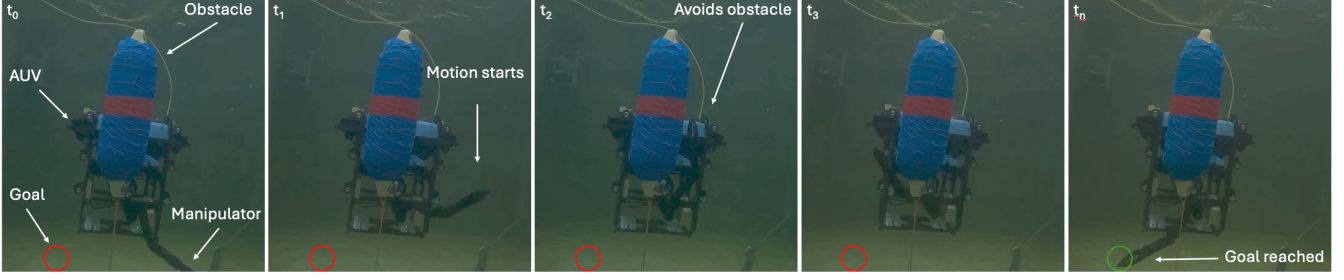


Fig. 3: The AI agent workflow for the UVMS manipulation task. A high-level textual command from the *User terminal* is interpreted by the *Commander AI Agent*, which then directs the *Motion Planning* and *Perception* agents to execute the task.

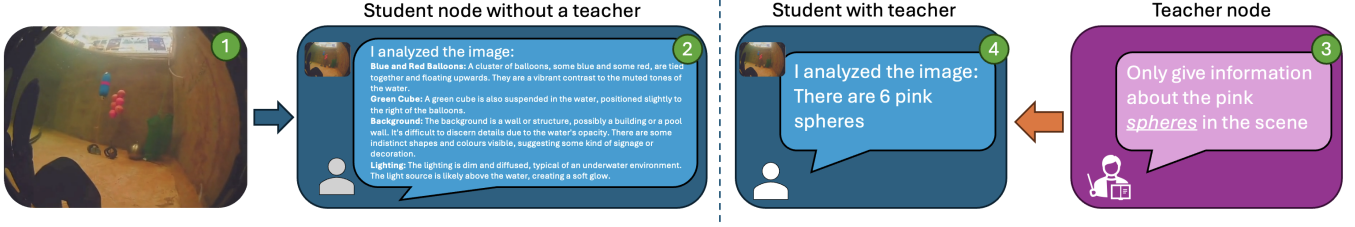


Fig. 4: A direct *Before and After* comparison of the Teacher-Student instructional tuning mechanism. (Left) Without the *Teacher* enabled, the *Student* agent observes the scene (1) and defaults to a verbose scene description as seen in (2). (Right) With the *Teacher* enabled, the *Teacher* gives precise feedback (3) to the *Student* over several episodes, resulting in a converged output that is concise (4) and provides only the information requested by the *Teacher*

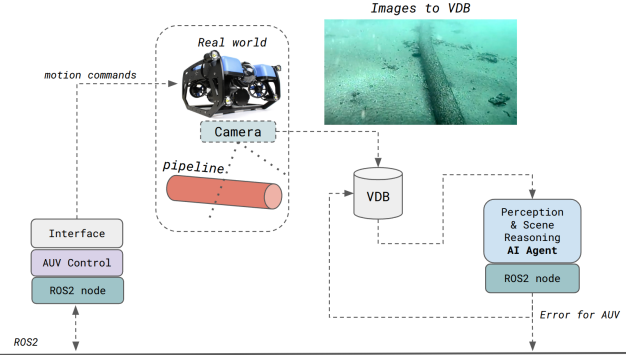


Fig. 5: The control loop for visually-guided disturbance rejection. The AUV's camera feeds a continuous stream of images into a VDB, which provides visual-temporal memory. The *Perception & Scene Reasoning AI Agent* uses this historical context to calculate a precise positional error, which the AUV's control system then uses to issue corrective motion commands.

natural language requirements for a new node. A trial was considered a *success* only if the agent completed the full end-to-end process: correctly interpreting the request, generating valid Python code, creating and passing its own unit tests, and integrating the new node into the live system. A *failure* was defined as any case where the generated code did not compile, failed its unit tests, or was functionally incorrect.

We conducted 10 trials for each of the following field-relevant scenarios: The field-relevant scenarios included generating: (1) a stateful noise-reduction filter to average sensor values over a 10-sample window; (2) a standard sensor fusion

TABLE VI: Performance of Autonomous Node Generation

Generated Node	Avg. Gen. Time (s)	Success Rate
Stateful Averaging Filter	1.5	80%
Kalman Filter (2x Odom)	5.8	80%
Kalman Filter (DVL+Compass)	5.9	70%

node implementing a Kalman filter for two odometry topics; and (3) an emergency navigation repair node. This primary evaluation scenario simulated a critical Inertial Navigation System (INS) failure, requiring the system to synthesize a new Kalman filter to fuse compass and Doppler Velocity Log (DVL) data to mitigate navigational drift.

The performance of the autonomous generation process is summarized in Table VI. Most significantly, in the navigation failure scenario, the dynamically generated Kalman filter node, once integrated, successfully fused the compass and DVL data, reducing the navigational drift rate by an estimated 70% compared to dead-reckoning with DVL alone. This provides powerful evidence of the system's ability to perform runtime self-repair of a critical capability.

Most significantly, in the navigation failure scenario, the dynamically generated Kalman filter node, once integrated, successfully fused the compass and DVL data, reducing the navigational drift rate by an estimated 70% compared to dead-reckoning with DVL alone.

C. Dynamic, Predictive System Diagnostics

This capability was evaluated through a series of proof-of-concept experiments where we simulated various hardware failure modes. For each test case, we programmatically disabled

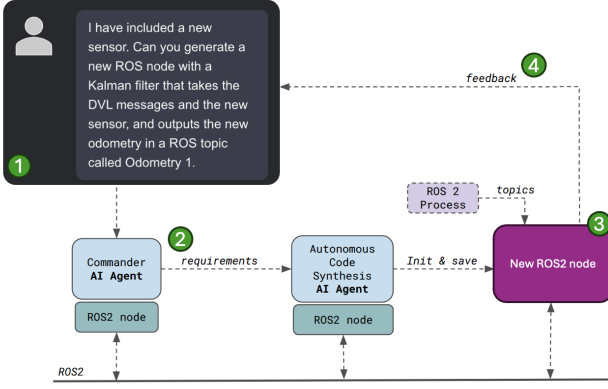


Fig. 6: The evaluation workflow for autonomous node generation. 1) A user gives a query. 2) The *Commander AI Agent* generates the requirements and passes them over to the *Autonomous Code Synthesis Agent*. 3) The *Autonomous Code Synthesis Agent* generates, tests, and deploys the new ROS 2 node. 4) For this evaluation, feedback on the new node’s performance was routed to the user terminal. It is important to note that in a fully autonomous mission, the *Commander AI Agent* can make the starting request and the feedback would be sent back to it to inform subsequent decisions.

```
class OdometryFusionNode(Node):
    def __init__(self):
        super().__init__('odometry_fusion_node')

        # Subscribers
        self.subscription_1 = self.create_subscription(Odometry,
            '/odometry_1', self.odometry_1_callback, 10)
        self.subscription_2 = self.create_subscription(Odometry,
            '/odometry_2', self.odometry_2_callback, 10)
        # Publisher
        self.publisher_3 = self.create_publisher(Odometry, '/odometry_3', 10)
        # Kalman Filter initialization
        self.Q = np.diag([1e-6, 1e-6, 1e-6]) # Process noise covariance
        self.R = np.diag([1e-2, 1e-2, 1e-2]) # Measurement noise covariance
        self.x_hat = np.zeros(3) # Initial state estimate
        self.P = np.eye(3) # Initial error covariance

    def kalman_filter(self, measurement):
        # Prediction step
        x_hat_pred = self.x_hat
        P_pred = self.P + self.Q
        # Update step
        K = P_pred @ np.linalg.inv(P_pred + self.R)
        self.x_hat = x_hat_pred + K @ (measurement - x_hat_pred)
        self.P = (np.eye(3) - K) @ P_pred
        # Publish fused odometry
        fused_odom = Odometry()
        fused_odom.pose.pose.position.x = self.x_hat[0]
        fused_odom.pose.pose.position.y = self.x_hat[1]
```

Fig. 7: Excerpt of the code generated. For full details please check the website and videos.

specific thrusters via the vehicle’s control interface. The ArduPilot system [61], observing the lack of response, would then generate an updated vehicle status *JSON* reflecting the fault. The Diagnostic AI Agent’s task was to monitor this stream and correctly identify which thrusters were malfunctioning. We ran 5 trials for each fault configuration, and the agent correctly diagnosed the system state with 100% accuracy across all tests, as summarised in Table VII. To further demonstrate the agent’s ability to track a dynamically changing system state, we

TABLE VII: Summary of Diagnostic Test Cases and Results

Test Case	Simulated Fault	Diagnosis Accuracy
1	All Thrusters OK	100% (5/5)
2	Thruster 2 Disabled	100% (5/5)
3	Thruster 6 Disabled	100% (5/5)
4	Thrusters 2 & 3 Disabled	100% (5/5)
5	Thrusters 2, 3, 6, 7 Disabled	100% (5/5)

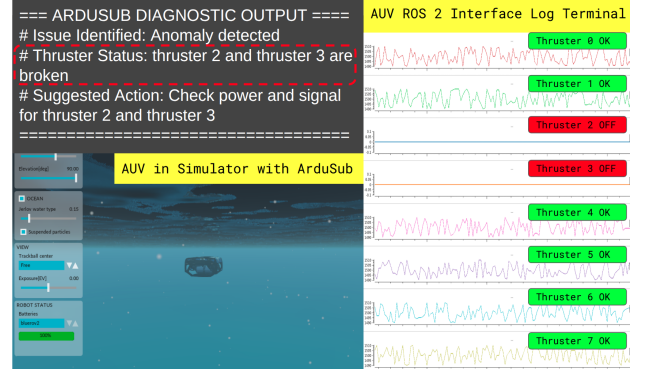


Fig. 8: Validation of the Predictive Diagnostics Agent during a simulated two-thruster failure. The plots on the right show the raw PWM signals from the ROS 2 interface, clearly indicating that thrusters 2 and 3 are unresponsive (flatlined) while the others are active. The Diagnostic Agent continuously analyzes this time-series data, reasons about the discrepancy between commanded and actual behavior, and autonomously generates the correct high-level diagnosis shown in the terminal output (top left). This demonstrates the agent’s ability to translate low-level signal anomalies into a precise, human-readable fault report without relying on pre-programmed error codes.

conducted a specific transition test. First, we disabled thrusters 2 and 3 to induce a fault. After the agent correctly diagnosed this failure, we re-enabled the thrusters in the interface. The agent’s subsequent report immediately and accurately reflected the system’s return to a healthy state. Figure 8 shows the agent’s real-time terminal output, capturing these precise system faults.

D. Inherent Safety and Control Mechanisms

To validate the safety of the agent, we conduct a complex evaluation of multi-agent behaviour. Two AUVs, each controlled by an independent AI agent, navigated a maze-like environment toward a shared goal (Fig. 9). The agents must then coordinate to avoid each other, but must do so via emergent coordination, without preplanning. The key to this capability was embedding not only a negotiation protocol but also the vehicle’s physical dimensions into each agent’s behavioural constitution (SYSTEM prompt). This allowed each agent to reason spatially about its own footprint and the other’s intended path.

The evaluation involved 8 test runs with increasingly complex trajectory conflicts, where paths crossed or required agents to pass each other in tight corridors. Each test case was run 5 times to ensure repeatability. During operation, the agents continuously share their intended trajectories via ROS 2. Each

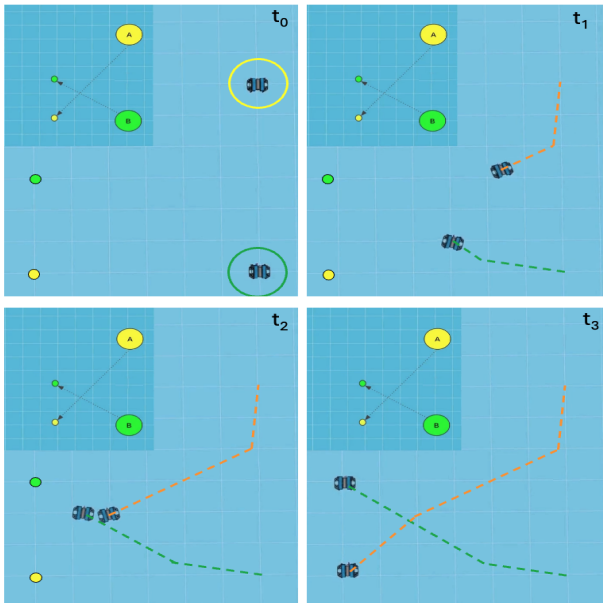


Fig. 9: Illustration of the decentralised collision avoidance scenario, where two AI agents negotiate a solution via ROS 2. Agent A, marked as Yellow, and Agent B, marked with green. Each has a starting position and a goal point, with conflicting goals.

TABLE VIII: Performance of Decentralised Negotiation (Avg. over 5 runs)

Test ID	Negotiation Time (s)	Min. Safe Distance (m)
1	0.05	1.2
2	0.05	1.4
3	0.06	2.1
4	0.12	0.21
5	0.13	0.22
6	0.06	2.32
7	0.14	0.32
8	0.13	0.76

agent then predicts the future states along both paths, calculating the minimum distance between their vehicle’s bounding volumes. If this predicted distance fell below a predefined safety threshold, a collision was considered imminent, triggering a human-independent negotiation dialogue. Based on its pre-configured reasoning process, one agent would identify the conflict and autonomously compute a new path yielding to the other. Table VIII summarises the averaged performance, focusing on the efficiency of the negotiation and the safety of the resulting manoeuvre.

Across all 40 trials, the agent pair successfully negotiated a collision-free path every time. The data in Table VIII shows that the negotiation was exceptionally efficient, with a resolution consistently found in well under 0.2 seconds. It should be noted that this time is presented as a reference, as it is dependent on the specific AI model and GPU hardware used. Critically, even in the most constrained scenarios where the vehicles had to pass with very tight clearances (e.g., a minimum distance of just 0.21 m in Test ID 4), a positive safe distance was always maintained. This consistent success in difficult configurations confirms that an effective, emergent deconfliction strategy can

be derived directly from high-level textual instructions about vehicle geometry.

VI. CONCLUSION

This paper presented UROSA, a distributed AI agent architecture that replaces monolithic programmatic control with a collaborative ecosystem of agentic entities. By embedding a structured reasoning process into each agent via behavioural constitutions, UROSA decouples high-level mission goals from low-level code implementation, enabling a new level of cognitive flexibility in the underwater domain. Through empirical validation, we demonstrated that this framework effectively handles multi-robot coordination from abstract inputs, improves performance through experiential learning via RAG, autonomously synthesises new software modules at runtime, and performs predictive diagnostics without static fault trees. These capabilities, protected by a multi-layered safety strategy, proved both reliable and effective across demanding scenarios.

While these findings are significant, challenges in deploying such cognitive systems remain. The engineering of effective behavioural constitutions is a complex new skill, and the formal verification and real-time performance of every AI-driven decision require further research. Future work will address these challenges by exploring two frontiers: automating the generation of agent constitutions, and enabling runtime self-reconfiguration. This latter capability would allow an agent, upon receiving a diagnosis of hardware failure, to autonomously rewrite and deploy a new control model—such as a modified thruster allocation matrix—to ensure the system continues its mission with gracefully degraded performance. This research provides a scalable and safe framework for developing more adaptable and resilient autonomous systems, moving beyond pre-programmed tools towards reasoning partners capable of overcoming both environmental and internal, unforeseen challenges.

REFERENCES

- [1] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*. Pearson Education, 2020.
- [2] N. J. Nilsson, “Shakey the robot,” SRI International Menlo Park CA Artificial Intelligence Center, Tech. Rep., 1984.
- [3] R. A. Brooks, “Intelligence without representation,” *Artificial Intelligence*, vol. 47, no. 1-3, pp. 139–159, 1991.
- [4] R. C. Arkin, *Behavior-based robotics*. MIT Press, 1998.
- [5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, others, and D. Amodei, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, others, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [7] J. E. Laird, *The Soar cognitive architecture*. MIT Press, 2012.
- [8] J. R. Anderson, E. H. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, “An integrated theory of the mind,” *Psychological Review*, vol. 111, no. 4, p. 1036, 2004.
- [9] Z. Ji, N. Lee, R. Fries, T. Yu, D. Su, Y. Xu, others, and A. Madotto, “Survey of hallucination in natural language generation,” *ACM Computing Surveys*, 2023.
- [10] L. Huang, D. Y. Zhou, H. Edwards, and W. Zhou, “A survey on hallucination in large language models,” arXiv preprint arXiv:2305.13565, 2023.

- [11] R. Royce, M. Kaufmann, J. Beckett, S. Moon, K. Carpenter, K. Pak, A. Towler, R. Thakker, and S. Khattak, "Enabling novel mission operations and interactions with rosa: The robot operating system agent," NASA/JPL-Caltech Technical Report, 2025.
- [12] Anthropic, "Alignment research," <https://www.anthropic.com/research/alignment>, 2024.
- [13] OpenAI, "Alignment and safety," <https://openai.com/safety>, 2024.
- [14] EY, "The future of autonomous systems: agentic ai in consumer products," *EY Insights*, March 2025.
- [15] Endava, "Agentic ai: Definition, types, applications," 2025.
- [16] J. McCarthy, "Programs with common sense," in *Mechanisation of Thought Processes*, vol. 1, 1959, pp. 77–84.
- [17] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to theorem proving applied to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.
- [18] J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," *Machine Intelligence*, vol. 4, pp. 463–502, 1969.
- [19] P. E. Agre and D. Chapman, "Pengi: An implementation of a theory of activity," in *AAAI*, 1987, pp. 268–272.
- [20] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.
- [21] R. Pfeifer and C. Scheier, *Understanding intelligence*. MIT Press, 1999.
- [22] A. Clark, *Being there: Putting brain, body, and world together again*. MIT Press, 1997.
- [23] M. Wilson, "Six views of embodied cognition," *Psychonomic Bulletin & Review*, vol. 9, no. 4, pp. 625–636, 2002.
- [24] J. R. Anderson, *How can the mind occur in the body?* Oxford University Press, 2007.
- [25] R. J. Brachman and H. J. Levesque, "The tractability of subsumption in frame-based description languages," in *AAAI*, 1984, pp. 34–37.
- [26] P. Kumar, "Large language models (LLMs): survey, technical frameworks, and future challenges," *Artif. Intell. Rev.*, vol. 57, no. 10, pp. 1–51, Aug. 2024.
- [27] OpenAI, "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023.
- [28] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, F. Kamani, others, and Y. Zhang, "Sparks of artificial general intelligence: Early experiments with gpt-4," arXiv preprint arXiv:2303.12712, 2023.
- [29] N. Mirchev, S. Jiang, S. Shah, and A. Garg, "Large language models as general-purpose policies for robots," arXiv preprint arXiv:2305.05042, 2023.
- [30] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, G. Danihelka, others, and R. Tanwani, "Do as i can, not as i say: Grounding language in robotic affordances," arXiv preprint arXiv:2204.01691, 2022.
- [31] A. Brohan, N. Brown, W. L. Brohan, Y. C. Chen, Y. Chebotar, J. Castro, others, and S. Levine, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," arXiv preprint arXiv:2207.05736, 2023.
- [32] A. Radford, J. W. Kim, C. Xu, G. Xu, G. Brockman, others, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.
- [33] A. K. Gupta, A. Yan, J. Guo, Y. Cheng, L. Yang, L. H. Chen, others, and K. Keutzer, "Visual grounding for language-guided navigation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3562–3572.
- [34] D. Driess, A. Zeng, S. Cabi, M. S. Sajjadi, D. Allen, C. Lynch, others, and M. Toussaint, "Palm-e: An embodiment-aware language model for instruction following with visual and tactile feedback," arXiv preprint arXiv:2203.16939, 2023.
- [35] C. Jia, Y. T. Chen, Z. Lu, S. Tunyasuvunakool, N. De Freitas, and D. Tarlow, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 4615–4625.
- [36] K. Zhu, C. Gan, L. Wang, Y. Fang, X. Dai, and S. Han, "Vision-language models are zero-shot reward function approximators," arXiv preprint arXiv:2303.02896, 2023.
- [37] J. Liang, W. Zeng, G. Mu, S. Yang, T. L. Griffiths, Y. Zhu, others, and D. Song, "Code as policies: Language model-based discrete action policies for embodied ai," arXiv preprint arXiv:2209.07753, 2023.
- [38] A. Zeng, S. Song, C. Lee, N. Rodriguez-Ruiz, T. Van-Hove, R. S. Fearing, others, and C. Lynch, "Socratic models: Composing zero-shot multimodality with language," in *International Conference on Machine Learning*. PMLR, 2023, pp. 26 944–26 967.
- [39] S. Yao, W. Zhao, J. Wang, Y. Cao, S. Narasimhan, and D. Zhao, "React: Synergizing reasoning and acting in language models for task solving," arXiv preprint arXiv:2210.03629, 2023.
- [40] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, no. 1, pp. 7–27, 1997.
- [41] P. Stone, Ed., *Multiagent systems: A modern approach to distributed artificial intelligence*. MIT Press, 2000.
- [42] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.
- [43] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.
- [44] J. M. Bradshaw, R. R. Hoffman, M. Johnson, and P. J. Feltovich, "Beyond human-centered autonomy: Collaboration and reciprocal adaptation in human-agent teams," *IEEE Intelligent Systems*, vol. 32, no. 3, pp. 70–78, 2017.
- [45] M. Longo, L. Rathenau, and J. Weber, *Ethical autonomy in intelligent agents: Embedding ethics into artificial intelligence*. Springer Nature, 2023.
- [46] R. D. Beer, M. Randall, and P. Fitch, "Evolving dynamical neural networks for adaptive behavior," *Adaptive Behavior*, vol. 22, no. 1, pp. 3–27, 2014.
- [47] J. Maynez, S. Narayan, L. Lokhande, and R. Reddy, "On faithfulness and hallucination in abstractive summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 1883–1896.
- [48] M. Verhelst and E. Moons, "What edge computing can do for deep learning," in *2017 IEEE International Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2017, pp. 1–6.
- [49] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and I. Sutskever, "Concrete ai safety problems," arXiv preprint arXiv:1606.06565, 2016.
- [50] P. Koopman and M. Wagner, "Challenges in autonomous vehicle verification and validation," *SAE International Journal of Transportation Safety*, vol. 5, no. 1, pp. 19–27, 2017.
- [51] P. Lin, K. Abney, and R. Jenkins, "Robot ethics: Mapping the issues for computer scientists," *AI Magazine*, vol. 32, no. 1, p. 15, 2011.
- [52] W. Wallach and C. Allen, *Moral machines: Teaching robots right from wrong*. Oxford University Press, 2008.
- [53] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," arXiv preprint arXiv:2201.11903, 2022.
- [54] A. Madaan, N. Tandon, P. Gupta, K. Hallinan, L. Gao, S. Wiegrefe, U. Alon, P. Cair, Y. Yang, A. Cohan *et al.*, "Self-refine: Iterative refinement with self-feedback," arXiv preprint arXiv:2303.17651, 2023.
- [55] J. S. Willners, I. Carlucho, S. Katagiri, C. Lemoine, J. Roe, D. Stephens, T. Łuczyński, S. Xu, Y. Carreno, É. Pairet *et al.*, "From market-ready rovs to low-cost auvs," in *OCEANS 2021: San Diego–Porto*. IEEE, 2021, pp. 1–7.
- [56] M. Buchholz, I. Carlucho, Z. Huang, M. Grimaldi, P. Nicolay, S. Tunçay, and Y. R. Petillot, "Framework for robust motion planning of tethered multi-robot systems in marine environments," in *Proceedings of the IEEE/MTS OCEANS Conference*. Brest, France: IEEE, May 2025.
- [57] M. Buchholz, I. Carlucho, M. Grimaldi, and Y. R. Petillot, "Tethered multi-robot systems in marine environments," in *Proceedings of the ICRA 2025 Workshop on Marine Robotics*, New Orleans, USA, 2025.
- [58] M. Buchholz, I. Carlucho, M. Grimaldi, M. Koskinopoulou, and Y. R. Petillot, "Context-aware behavior learning with heuristic motion memory for underwater manipulation," 2025. [Online]. Available: <https://arxiv.org/abs/2507.14099>
- [59] E. Morgan, I. Carlucho, W. Ard, and C. Barbalata, "Autonomous underwater manipulation: Current trends in dynamics, control, planning, perception, and future directions," *Current Robotics Reports*, vol. 3, no. 4, pp. 187–198, 2022.
- [60] M. Buchholz, I. Carlucho, M. Grimaldi, and Y. R. Petillot, "Context-aware behavior learning with heuristic motion memory for underwater manipulation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. New Orleans, USA: IEEE, 2025.
- [61] The ArduPilot Development Team, "ArduPilot: Open source autopilot software suite," <https://ardupilot.org>, 2025, accessed: 5-July-2025.