PROCEEDINGS

26th International Workshop on Physical Agents

September 28–29, 2026

Residencia V Centenario

Jarandilla de la Vera, Cáceres, Spain

Universidad de Extremadura

University of Extremadura · Spain

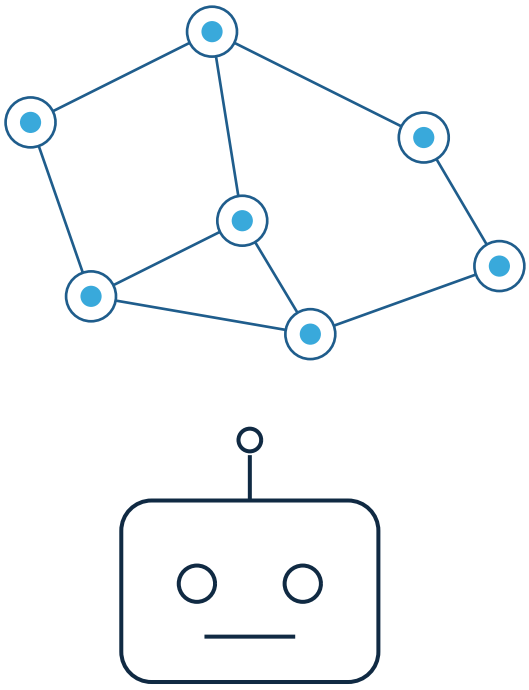


TABLE OF CONTENTS

WAF 2026 · 26th International Workshop on Physical Agents · Proceedings

VLN-Pilot: Large Vision-Language Model as an Autonomous Indoor Drone Operator

Bessie Dominguez-Dager*[0000−0003−2214−3849], Sergio Suescun-Ferrandiz[0009−0008−1521−0031], Felix Escalona[0000−0003−2245−601X], Francisco Gomez-Donoso[0000−0002−7830−2661], and Miguel Cazorla[0000−0001−6805−3633]

University Institute for Compute Research. University of Alicante.
Ctra. San Vicente del Raspeig SN, 03690, Alicante. SPAIN.
{bessie.dominguez, sergio.suescun, felix.escalona, fgomez, miguel.cazorla}@ua.es
*Corresponding Author

Abstract. This paper introduces **VLN-Pilot**, a novel framework in which a large Vision Language Model (VLM) assumes the role of a human pilot for indoor drone navigation. By leveraging the multimodal reasoning abilities of VLMs, VLN-Pilot interprets free-form natural language instructions and grounds them in visual observations to plan and execute drone trajectories in GPS-denied indoor environments. Unlike traditional rule-based or geometric path-planning approaches, our framework integrates language-driven semantic understanding with visual perception, enabling context-aware flight behaviors with minimal task-specific engineering. VLN-Pilot supports fully autonomous instruction-following by reasoning about spatial relationships, obstacle avoidance, and reactivity to unforeseen events. We validate our framework on a custom photorealistic indoor simulation benchmark, where the VLM-driven agent achieves high success rates on complex tasks, including long-horizon navigation with multiple semantic targets. These results highlight the promise of replacing remote pilots with a language-guided autonomous agent, enabling scalable, human-friendly control of indoor UAVs for applications such as inspection, search-and-rescue, and facility monitoring. The code is available at: https://github.com/bdager/drone-llm.

Keywords: Vision-and-Language Navigation · VLMs · UAV Autonomy · Indoor Drone Navigation · Natural Language Instruction Following

1 Introduction

Vision-and-Language Navigation (VLN) has emerged as a promising paradigm for enabling embodied agents to interpret natural language instructions and navigate complex environments by grounding actions in visual observations. While significant advances have been made in ground-based navigation, extending VLN to autonomous drones remains an open challenge. Drones operating indoors face

unique difficulties: constrained spaces, dynamic obstacles, absence of GPS, making human-in-the-loop piloting standard practice in industrial applications.

In this work, we introduce **VLN-Pilot**, a novel framework in which large Vision-Language Models (VLMs), such as GPT and Gemini, assume the role traditionally filled by a remote human pilot. By leveraging these models' reasoning abilities to interpret visual inputs and make high-level control decisions, VLN-Pilot aims to replace or augment the pilot. The drone is given only a topological map of the environment, describing the room connectivity graph, along with frontal images captured during flight. In this constrained sensory setting, the VLM must reason about its location, interpret language-driven mission goals, and plan semantic trajectories.

For low-level control, we employ a state-machine architecture that manages stable flight, obstacle avoidance, and basic drone commands, while the VLM acts as a supervisory controller, deciding when to transition between states based on its understanding of the navigation objective and the visual scene. This hybrid design yields interpretable, robust, and human-aligned behavior while benefiting from the high-level reasoning of modern VLMs.

We evaluate our prototype in a Unity-based simulation replicating the DJI Tello drone in realistic indoor environments, providing a safe and flexible testbed for complex instruction-following scenarios. The results show that VLN-Pilot significantly reduces human workload while maintaining effective, safe navigation, opening promising avenues for scalable, user-friendly indoor drone autonomy.

2 Related Work

VLN sits at the intersection of computer vision, natural language processing, and embodied robotics. Anderson et al. [1] laid the foundation with the Room-to-Room (R2R) benchmark, pairing a photorealistic simulator with natural language instructions; surveys [14] have since categorized the field into goal- and route-oriented tasks, alongside dialog-based settings.

Later work advanced the learning: Episodic Transformer [9] tracked full episode history for long-horizon tasks such as ALFRED [12], FlexVLN [15] combined hierarchical reasoning with LLMs for cross-dataset adaptation, SOAT [7] used scene-object-aware transformers to ground both coarse and fine-grained references, and VLN-R1 [10] showed that reinforcement fine-tuning with large VLMs enables end-to-end reasoning from video streams rather than static graphs.

Indoor drone navigation further requires 3D path planning, altitude control, and collision avoidance. UAV-VLN [11] pairs LLMs with visual perception for spatial reasoning in UAV control, AerialVLN [5] contributes a UAV dataset with a continuous, city-scale simulator, and Airbert [2] improves few-shot indoor performance via domain-adapted pretraining on mined path-instruction pairs. Recent work pushes toward greater realism and continuous control: VLFly [16] generates continuous velocity commands directly from monocular egocentric observations, avoiding discrete action spaces altogether, while TRAVEL [13] combines multi-view perception, realistic 6-DoF flight dynamics, and an assistant-guided

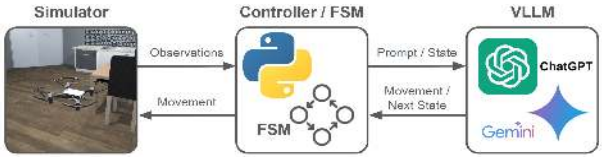


Fig. 1: Closed-loop pipeline between: Unity simulator, Python controller, VLM.

benchmark for collision-aware aerial object search. Closest to our setting, IndoorUAV [6] benchmarks indoor UAV navigation across over 1,000 continuous scenes, decomposing long-horizon missions into short executable sub-tasks handled by a multimodal agent. Complementary to per-step querying, VLMaps [3] fuses VLM features into a persistent semantic map supporting open-vocabulary spatial queries and robot-specific obstacle maps for safer path planning.

While the field moves toward training-heavy, continuous aerial navigation pipelines, VLN-Pilot takes the opposite route: no fine-tuning and no pre-built map, just an off-the-shelf VLM acting as an interpretable supervisory pilot over a lightweight FSM to replace a remote human operator with minimal engineering.

3 Methodology

VLN-Pilot combines VLMs with a rule-based finite-state machine (FSM) for autonomous indoor drone navigation, operating within a Unity-based simulator of a DJI Tello drone via the ML-Agents [4] framework. The pipeline has three modules: the Unity simulator provides visual observations and topological state information to a Python controller, which forwards these to a VLM (e.g., GPT, Gemini) through a structured prompt asking it to interpret the situation and output a high-level movement command and the next FSM state; the controller then parses the response, sends control signals to move the drone, and repeats the cycle (Figure 1). This closed loop lets the VLM act as a high-level decision-maker while the FSM handles low-level execution.

Unity Simulator. The Unity Simulator provides the virtual environment where the drone agent operates and is tested, modeling realistic indoor scenes with basic textures, lighting, and physical interactions. We simulate the DJI Tello drone, including its camera view, flight dynamics, and collision detection, to approximate real-world conditions. Using the ML-Agents toolkit, the simulator sends the following observations to the controller: a frontal RGB image, a rear RGB image to visualize the drone's behavior, the drone's position in X, Y, and Z coordinates, its orientation around the vertical axis (yaw), and its collision status, computed via Unity's physics engine. In response, the controller returns a motion command specifying movement along the left–right, front–back, and up–down axes, as well as rotation to the left or right (Figure 2 left). The indoor environment is based on the Furnished Cabin asset by NatureManufacture [8] from the Unity Asset Store, a furnished cabin with multiple connected rooms: a combined living room and kitchen, a bedroom, and a bathroom (Figure 2 right).

Fig. 2: Simulation overview: **Left** shows the drone platform with front/rear RGB views (axes: X=red, Y=green, Z=blue), and **Right** shows the furnished cabin layout (living room left, bedroom center, bathroom right).

Table 1: States of the drone navigation and interaction system.

State	**Description**
Start	Initial state; the system is activated.
Recognize room	Determines the room the drone is currently in.
Search open door	Searches for an open door to cross (target room $\neq$ current one).
Orient towards door	Aligns the drone to face a detected open door.
Go through door	Moves the drone through the oriented door.
Stay on room	*(Final state)* Target room reached, no object to find.
Search object	Searches for the target object within the room.
Reach object	Guides the drone closer to the object.
Describe object	*(Final state)* Describes the target's position and orientation.

Python Controller. The Python controller coordinates communication between the simulator and the VLM, launching experiments, managing the FSM that defines the drone's behavior, and checking whether the target has been reached. The FSM establishes a sequence of semantic states guiding the drone through navigation and interaction tasks (detailed in Table 1). To transition between states, the controller queries the VLM for the next state given the current visual input, current state, and previous movement; Figure 3 shows the complete FSM with the conditions used to decide each change. The environment is represented as a topological map: rather than precise metric coordinates, a graph whose nodes are rooms (e.g., living room, bedroom, bathroom) and edges are navigable connections (e.g., bedroom to living room). Finally, the controller uses predefined motion commands: forward (A1–A3: 10/25/50 cm), rotate right (B1–B3: 15°/45°/90°), rotate left (C1–C3: 15°/45°/90°), lateral (D1, D2: 10 cm left/right), and no movement (E).

VLM Module. This module builds prompts for the chosen VLM based on the drone's current state and observations, in three parts. First, a general description (shared across states) explains the model's role as a drone pilot navigating safely to a target, the input it receives, and how to structure its response. Second, a state-dependent part gives the current state's goal and rules, the movement commands allowed, and the possible next states. The model's input is the user's query, the frontal image, the topological map, and the previous state and move-

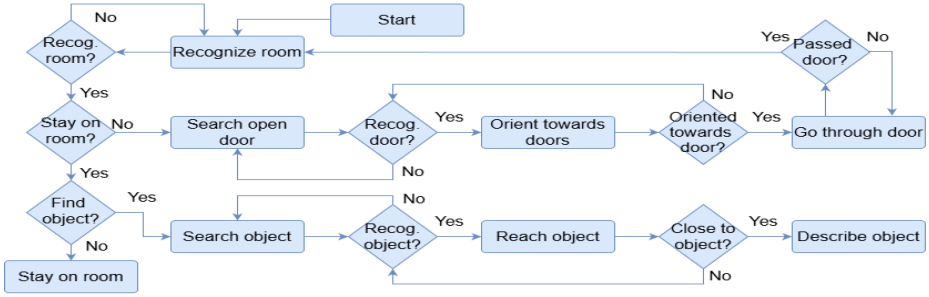


Fig. 3: FSM diagram of the drone controller: execution states (rectangles) and transition conditions (diamonds) evaluated via VLM queries.

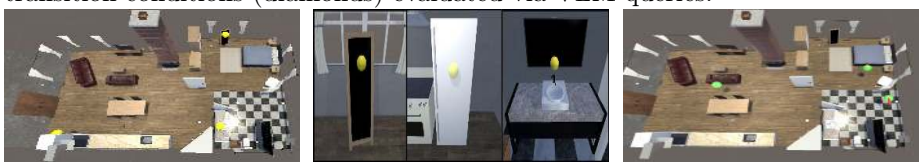


Fig. 4: Target positions (left), camera views (center), and spawn points (right): the green sphere marks the starting position, the red bar the initial orientation.

ment; from these it determines the next movement, updates the state, identifies the room it believes it is in, briefly describes the visual input, and explains its reasoning. Finally, a closing part specifies the output format.

4 Experimental Setup

Simulation. The system's main objectives are to navigate to a specified room and recognize relevant objects from the user query. The drone's room is determined from its X and Z coordinates (the Unity simulator's non-vertical axes). The possible targets are a refrigerator in the living room, a mirror in the bedroom, and a sink in the bathroom (Figure 4), and we define three spawn points for diverse initial conditions (Figure 4 right). The controller runs the simulation given the drone's initial position, the goal query, the number of repetitions (n=5), and the maximum allowed steps (m=50), monitoring whether the objective is reached by checking the drone's room and its proximity and orientation to a target. Execution ends when the target is reached, the model incorrectly believes it has, the drone collides with an obstacle, or the step limit is exceeded.

VLM server. We implemented a Flask web server to handle requests to the VLMs. Experiments used the `gpt-4.1` and `gemini-2.5-flash` models, accessed via the `openai` and `google-genai` Python libraries, respectively.

5 Results and Discussion

We evaluate the GPT and Gemini VLMs on our task. Table 2 summarizes five executions from the same spawn point and query, measuring goal achievement,

collisions, and whether the agent exceeded the maximum allowed steps. GPT outperformed Gemini in several cases, such as "Go to the bedroom" and "Find the refrigerator" from the living room/kitchen, and "Go to the living room/kitchen" from the bedroom, where Gemini more often exhausted the step limit and incurred more collisions, yielding less efficient navigation.

Table 2: Comparison of GPT and Gemini results grouped by starting room: achievement scores, collisions, and maximum steps exceeded.

Starting Room	**Query**	**GPT**			**Gemini**		
		Ach.	**Coll.**	**Steps**	**Ach.**	**Coll.**	**Steps**
Living Room and Kitchen	Go to the bathroom	2/5	3	–	2/5	1	2
	Go to the bedroom	4/5	–	1	0/5	1	4
	Go to the living room/kitchen	5/5	–	–	5/5	–	–
	Find the refrigerator	4/5	–	1	0/5	–	5
Bedroom	Go to the bedroom	5/5	–	–	5/5	–	–
	Go to the living room/kitchen	4/5	1	–	2/5	3	–
	Find the mirror	4/5	–	1	4/5	–	1
Bathroom	Go to the bathroom	5/5	–	–	5/5	–	–
	Go to the living room/kitchen	5/5	–	–	5/5	–	–
	Find the sink	5/5	–	–	5/5	–	–

GPT effectively leveraged the semantic graph and state transitions to guide the drone toward target rooms, correctly interpreting the room connectivity graph, selecting the appropriate doorway, and maintaining consistent orientation. In particular, it interpreted "being near the door" as requiring the drone to be almost directly adjacent before crossing, which aligned well with the simulation's proximity thresholds and yielded reliable target-room identification.

Gemini interpreted spatial relationships differently, treating "being near the door" as maintaining a larger distance and requiring the doorway to appear precisely centered in its field of view before crossing. This produced oscillatory recentering from distant positions that often prevented the drone from approaching closely enough to cross. Modifying Gemini's prompt to approach more closely before aligning reduced the oscillation but introduced a new problem: the drone collided with door frames and nearby obstacles during close-quarters maneuvering. This fragility points to a deeper limitation shared by both models: their notion of "centered" lacks the spatial awareness to determine whether the drone can physically fit through a doorway, causing it to clip door frames or propeller tips while believing it was sufficiently aligned. The VLMs thus operate under a simplified notion of alignment, without true volumetric awareness of the drone's dimensions relative to the available doorway clearance.

6 Conclusion and Future Work

We presented VLN-Pilot, a hybrid VLM–FSM framework for autonomous indoor drone navigation, and evaluated GPT and Gemini as high-level pilots.

GPT proved the more practical baseline, while Gemini's stricter centering criteria caused oscillation that prompt-based fixes could not fully resolve without introducing collisions. Both models lacked volumetric awareness of the drone's dimensions relative to doorway clearance, motivating future work on structured spatial constraints, refined semantic-state integration, and grounding in real-world drone perception pipelines.

References

1. Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Gould, S., van den Hengel, A.: Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In: CVPR (2018)
2. Guhur, P.L., Tapaswi, M., Chen, S., Laptev, I., Schmid, C.: Airbert: In-domain Pretraining for Vision-and-Language Navigation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)
3. Huang, C., Mees, O., Zeng, A., Burgard, W.: Visual language maps for robot navigation. In: ICRA (2023)
4. Juliani, A., Berges, V.P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D.: Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627 (2020), `https://arxiv.org/pdf/1809.02627.pdf`
5. Liu, S., Zhang, H., Qi, Y., Wang, P., Zhang, Y., Wu, Q.: Aerialvln: Vision-and-language navigation for uavs. In: ICCV (2023)
6. Liu, X., Liu, Y., Qiu, H., Qirong, Y., Lian, Z.: IndoorUAV: Benchmarking vision-language UAV navigation in continuous indoor environments. In: AAAI (2026)
7. Moudgil, A., Majumdar, A., Agrawal, H., Lee, S., Batra, D.: Soat: A scene- and object-aware transformer for vision-and-language navigation. In: NeurIPS (2021)
8. NatureManufacture: Furnished cabin (2017), `https://assetstore.unity.com/packages/3d/environments/urban/furnished-cabin-71426`, unity Asset Store
9. Pashevich, A., Schmid, C., Sun, C.: Episodic transformer for vision-and-language navigation. In: NeurIPS (2021)
10. Qi, Z., Zhang, Z., Yu, Y., Wang, J., Zhao, H.: Vln-r1: Vision-language navigation via reinforcement fine-tuning. arXiv preprint arXiv:2506.17221 (2025)
11. Saxena, P., Raghuvanshi, N., Goveas, N.: Uav-vln: End-to-end vision language guided navigation for uavs. In: 2025 European Conference on Mobile Robots (ECMR). pp. 1–6 (2025). https://doi.org/10.1109/ECMR65884.2025.11163198
12. Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., Fox, D.: ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
13. Wang, X., Yang, D., Wang, Z., Kwan, H., Chen, J., Wu, W., Li, H., Liao, Y., Liu, S.: Towards realistic UAV vision-language navigation: Platform, benchmark, and methodology. arXiv preprint arXiv:2410.07087 (2024)
14. Wu, W., Chang, T., Li, X., Yin, Q., Zhu, J.: Vision-language navigation: A survey and taxonomy. IEEE Transactions on Neural Networks and Learning Systems (2022)
15. Zhang, S., Qiao, Y., Wang, Q., Guo, L., Wei, Z., Liu, J.: Flexvln: Flexible adaptation for diverse vision-and-language navigation tasks. IEEE Transactions on Multimedia **27**, 6307–6318 (2025). https://doi.org/10.1109/TMM.2025.3581809

16. Zhang, Y., Yu, H., Xiao, J., Feroskhan, M.: Grounded vision-language navigation for UAVs with open-vocabulary goal understanding. arXiv preprint arXiv:2506.10756 (2025)

A Prompt Architecture and FSM Integration

The prompting system is dynamically orchestrated by a FSM. Rather than employing a single static prompt, the input sent to the VLM at each control step t is assembled on-the-fly by combining general operational guidelines with state-specific execution rules.

Prompt Composition. The complete prompt constructed in each control loop consists of three modular components. A *global system prompt* defines the robot assistant persona, specifies the contextual inputs provided, and outlines the high-level operational objectives. *Dynamic FSM state rules* then inject the active state's specific goal, action policy, allowed atomic commands, and valid transition logic (e.g., *Search Object*, *Reach Object*, *Search Open Door*). Finally, *output format constraints* enforce strict single-JSON responses, ensuring deterministic parsing and preventing natural language conversational output.

Model Inputs and Task Expectations. At each step t, the VLM is queried to serve as a high-level policy regulator, receiving five distinct inputs: the visual observation $\mathbf{I}_t$, an egocentric RGB image captured by the camera and Base64-encoded; the user query Q, a natural language target objective (e.g., *"Find the mirror"*); the topological map M, a JSON graph defining indoor room connectivity; the current state S_t, the active state label in the FSM; and the previous movement A_{t-1}, the last executed atomic action, included to prevent oscillatory loops.

Output Specification The VLM must respond exclusively with a valid JSON object. Table 3 outlines the structured fields returned by the model per iteration to update the controller.

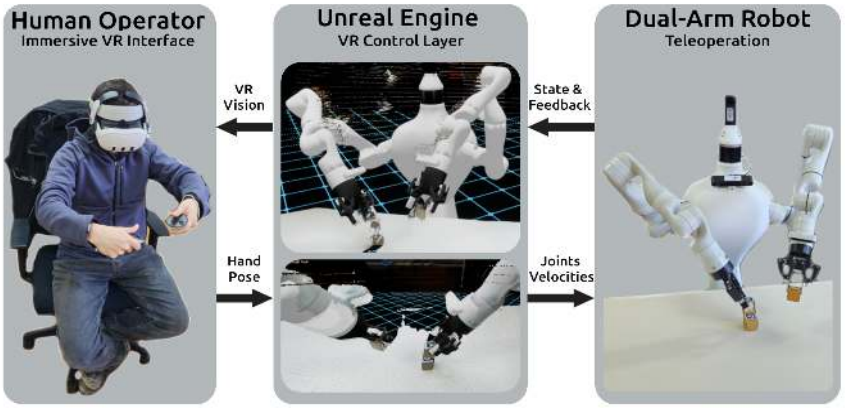
JSON Field	**Type**	**Description**
`room`	*string*	Identified room name from topological map M
`movement`	*string*	Single valid command allowed in S_t(e.g., `A1`, `B2`).
`state`	*string*	Predicted next FSM state (S_{t+1}).
`description`	*string*	Visual description and reasoning over observed landmarks.
`door_position`	*enum*	Relative door position

Table 3: VLM structured output schema mapping visual observations into FSM control signals.

Teleoperation of Dual-Arm Manipulators via VR Interfaces: A Framework Integrating Simulation and Real-World Control

Alejandro Torrejón[1 [0009−0001−7388−7429]], Sergio Eslava[1 [0009−0003−8674−9572]], Jorge Calderón[1 [0009−0003−6254−8506]], Pedro Nuñez[1 [0000−0002−3615−8833]] and Pablo Bustos[1 [0000−0003−3202−2493]]

RoboLab, Robotics and Artificial Vision, University of Extremadura, 10003 Cáceres, Spain



Abstract. We present a virtual reality (VR) framework for controlling dual-arm robotic manipulators through immersive interfaces, integrating both simulated and real-world platforms. The system combines the Webots robotics simulator with Unreal Engine 5.6.1 to provide real-time visualization and interaction, enabling users to manipulate each arm's tool point via VR controllers with natural depth perception and motion tracking. The same control interface is seamlessly extended to a physical dual-arm robot, allowing for teleoperation using the same VR environment. Our architecture supports real-time bidirectional communication between the VR layer and both the simulator and hardware, enabling responsive control and feedback. Performance evaluations demonstrate the viability of immersive VR as a unified interface for both simulation and physical control. The full journal paper is available at https://doi.org/10.3390/electronics15030572

Keywords: Virtual Reality; Teleoperation; Dual-arm robotic manipulators; Real-time visualization; Immersive interfaces

Improving on-edge LLM task planning for service robotics through LoRA fine-tuning

Alejandro González-Cantón[0009−0000−5729−0385], Miguel A. González-Santamarta[0000−0002−7658−8600], Irene González-Fernández[0009−0006−4350−2328], and Francisco J. Rodriguez-Lera[0000−0002−8400−7079]

Universidad de León, I4 Institute, Grupo de Robótica, Spain
algonzc@unileon.es

Abstract. Large Language Models (LLMs) have recently emerged as a promising alternative for robot task planning, enabling service robots to transform natural language instructions into executable action sequences. However, deploying LLM-based planners on edge devices remains challenging due to the limited capabilities of lightweight models, which frequently generate incomplete plans, hallucinated actions, or outputs that do not comply with the robot's execution framework. This paper presents a LoRA-based adaptation of *GPSR_planning*, a ROS 2 planning component designed for onboard task planning in RoboCup @Home General Purpose Service Robot (GPSR) scenarios. The proposed approach fine-tunes lightweight LLMs using command-plan pairs derived from robot-executable behaviors, allowing the model to better align generated plans with the robot's available capabilities while preserving offline execution and low computational requirements. The adapted models are evaluated using GPSR commands generated from the official RoboCup@Home benchmark. Experimental results show that LoRA fine-tuning significantly improves plan validity and action consistency, reducing hallucinated and non-executable behaviors compared to prompt-based planning alone. These findings demonstrate that lightweight domain adaptation can substantially increase the reliability of on-edge LLM planning without requiring cloud-based inference or modifications to the underlying model architecture.

Keywords: Robotics · Task planning · LLMs · Edge AI · LoRA

1 Introduction

Task planning is a core capability in service robotics, allowing robots to transform high-level goals into executable sequences of actions. Traditional approaches, such as symbolic planners[1], provide structured planning mechanisms, but they require detailed domain specifications and struggle to generalize to tasks not explicitly modeled in advance.

Recent progress in Large Language Models (LLMs) has opened new possibilities for robot task planning, particularly due to their ability to interpret

natural language instructions, reason over contextual information, and generate flexible action plans [4]. However, using LLMs in General Purpose Service Robot (GPSR) scenarios presents important challenges. Cloud-based models often depend on external APIs, while smaller onboard models may suffer from reduced accuracy, formatting errors, hallucinated actions, or limited adaptation to the robot's available capabilities.

This work extends *GPSR_planning* [2], a ROS 2 planning component for onboard service robots, by introducing Low-Rank Adaption (LoRA) fine-tuning [3] for GPSR task planning. The proposed approach trains a task-specific adapter using GPSR command-plan examples, allowing the planner to better adapt to the robot's executable behaviors while keeping the base LLM unchanged. This enables a modular planning system in which the GPSR specific adaptation can be activated when required, preserving the flexibility of on-edge LLM deployment.

2 System Context

The proposed work is integrated into the *CoreSense4Home* architecture, developed within the CORESENSE project [5]. As shown in Figure 1, the architecture is organized into four layers that separate robot control, high-level behaviors, system abstractions, and supporting components.

The *Control layer* coordinates task execution through behavior trees and manages the interaction between behaviors using a shared knowledge base. The *Behavior layer* defines the high-level actions the robot can perform, such as navigating to a location, searching for a person, detecting an object, or interacting with the user. These behaviors constitute the action space used by the planner to generate executable plans.

The *Systems layer* provides abstractions over the robot capabilities and AI modules, including perception, navigation, dialogue, manipulation, and planning. Within this layer, the *Planning system* receives a natural language command and generates a structured action plan using an onboard LLM. The generated plan is then interpreted by the control layer and executed through the corresponding robot behaviors. Finally, the *Components layer* contains the ROS 2 nodes, inference runtimes, models, libraries, and configuration files required by the upper layers.

3 Experimental Setup

The evaluation is carried out in a ROS 2 Humble environment using the TiAGo service robot in the Leon@Home Testbed, a certified European Robotics League facility that reproduces realistic domestic scenarios. The system integrates speech recognition, text-to-speech, object detection, navigation, behavior execution, and LLM-based planning.

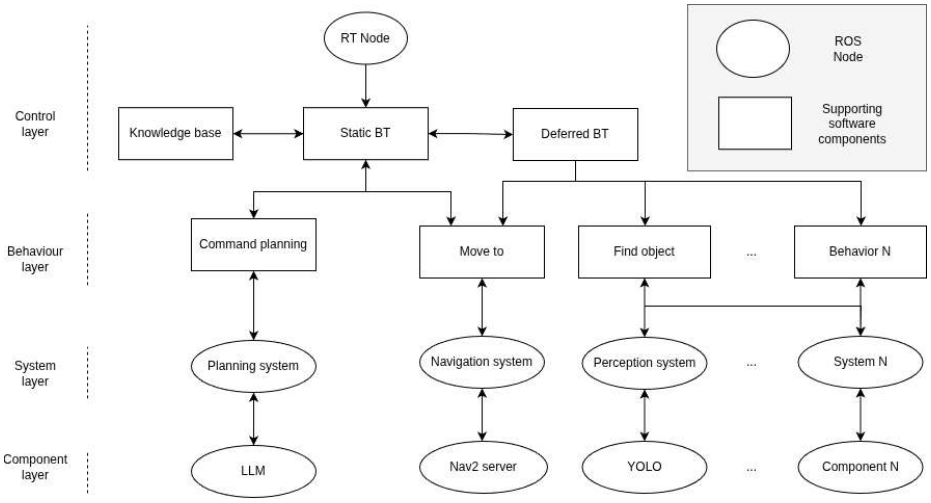


Fig. 1. System architecture showing the integration of the Planning System within the robot cognitive architecture.

The planner is deployed through *llama_ros*, enabling offline execution of lightweight LLMs. The baseline system relies on prompt engineering and JSON-schema-based constrained generation, while the proposed approach adapts the selected LLM using LoRA fine-tuning.

3.1 Planning Pipeline

The planning module receives a natural language command and generates a structured JSON plan composed of robot-executable behaviors, such as navigation, object search, person search, speaking, or object delivery. Each behavior is described through configuration files that specify its natural language description, accepted parameters, and usage constraints.

In the baseline system, the LLM is guided by a prompt containing the robot context, available behaviors, output format, and task constraints. The generated JSON plan is then converted into an XML behavior tree compatible with BehaviorTree.CPP, allowing the robot to execute the sequence of actions through the control layer.

3.2 LoRA Fine-Tuning and Evaluation

The evaluation considers lightweight LLMs suitable for onboard GPSR task planning, including Gemma E2B, Gemma E4B, Qwen3.5 4B, and Llama-3.2 3B. The models are adapted on an NVIDIA L4 GPU. The LoRA configuration uses a rank of $r = 32$, $\alpha = 16$, a learning rate of 2×10^{-4}, and two training epochs.

The dataset consists of 6000 GPSR command entries generated with the official RoboCup@Home command generator. For each command, a reference plan

is synthetically generated using Qwen2.5 72B and represented as a structured JSON plan composed only of robot-executable behaviors.

The adapted planners are executed on an NVIDIA RTX 3070 laptop GPU, reflecting the target deployment conditions used by the robot. The evaluation focuses on both planning quality and deployment feasibility. Planning quality is assessed using plan validity and exact match accuracy with respect to the reference plan. Deployment feasibility is evaluated through inference time and VRAM usage.

3.3 Results

Table 1 summarizes the experimental results comparing baseline and LoRA-adapted models across multiple metrics. The evaluation is organized into three groups: exact metrics (exact match accuracy and unoptimized plan rate), which capture strict string-level alignment with the reference plan; semantic metrics (plan validity and semantic similarity), which capture the structural and meaning-level quality of the generated plans; and performance metrics (inference time and GPU memory consumption), which capture deployment feasibility on edge hardware.

Model	**Exact**		**Semantic**		**Performance**	
Model	**Exact (%)**	**Unopt. (%)**	**Valid.**	**Sim.**	**Time (s)**	**VRAM (MiB)**
Gemma E2B baseline	13.40	15.00	0.693	0.758	3.02	2688
Gemma E2B LoRA	21.11	26.67	0.814	0.832	3.97	2845
Gemma E4B baseline	13.60	16.20	0.778	0.809	4.04	3872
Gemma E4B LoRA	36.76	38.92	0.884	0.901	4.80	4152
Llama-3.2 3B baseline	3.40	11.20	0.566	0.672	2.26	4188
Llama-3.2 3B LoRA	44.60	48.40	0.860	0.911	2.67	4190
Qwen3.5 4B baseline	12.20	21.20	0.702	0.764	4.43	5288
Qwen3.5 4B LoRA	53.00	54.80	0.937	0.897	4.06	5412

Table 1. Evaluation results comparing baseline and LoRA-adapted lightweight LLMs for GPSR task planning.

The results demonstrate that LoRA fine-tuning consistently improves plan validity and exact match accuracy across all four model families. The most notable gains are observed in Llama-3.2 3B and Qwen3.5 4B, where exact match accuracy increases from 3.40% to 44.60% and from 12.20% to 53.00%, respectively. Plan validity also improves substantially, with Qwen3.5 4B LoRA achieving the highest validity score (0.9375), closely followed by Gemma E4B LoRA (0.8843) and Llama-3.2 3B LoRA (0.8602).

Regarding time and resource constraints, inference time remains within acceptable limits for all adapted models, with mean planning times ranging from

2.67s to 4.80s. VRAM consumption increases only marginally with LoRA adapters, under 10% in all cases, as the additional adapter parameters are small relative to the base model, and remains within the capacity of the provided GPU.

Interestingly, the unoptimized plan rate tends to increase alongside exact match accuracy in LoRA-adapted models. This reflects a behavior shift where the fine-tuned model produces more structured outputs that are closer to the expected plan format, yet some outputs still contain minor structural inconsistencies that require post-processing. The overall trend confirms that domain-specific LoRA fine-tuning substantially improves the reliability of lightweight LLMs for on-edge GPSR task planning.

The proposed system was further validated during the RoboCup@Home Portugal 2026 competition, where it was deployed on the TiAGo robot within the official RoboCup@Home testbed environment. The LoRA-adapted planner was used in multiple GPSR executions throughout the competition, achieving the best performance in its category across several runs. This real-world deployment confirms that the static evaluation metrics translate into reliable plan generation under competition conditions, where the robot must interpret unseen natural language commands and execute the resulting plans in a dynamic environment with untrained users.

4 Conclusions

This work introduces a LoRA-based extension of *GPSR_planning*, an onboard LLM planner that generates structured robot action plans from natural language commands. The proposed adaptation improves the reliability of lightweight LLMs for GPSR task planning while preserving offline deployment and the modularity of the original system. The approach was validated both through quantitative metrics benchmarks and through real-world deployment at RoboCup@Home Portugal 2026.

4.1 Acknowledgments

This work was partially supported by Grant PID2024-161761OB-C21 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU. This work was also co-funded by the Consejería de Educación of the Junta de Castilla y León and the European Social Fund Plus (ESF+) through a predoctoral research staff contract grant.

References

1. Ghallab, M., Knoblock, C., Wilkins, D., Barrett, A., Christianson, D., Friedman, M., Kwok, C., Golden, K., Penberthy, S., Smith, D., Sun, Y., Weld, D.: PDDL - The Planning Domain Definition Language (Aug 1998)
2. González-Cantón, A., González-Santamarta, M.A., Rodriguez Lera, F.J., Guerrero-Higueras, Á.M., González-Fernández, I., Martín-Rico, F.: On-Edge Task Planning with Large Language Models for Service Robotics. In: Corchado, E., Quintián, H.,

Troncoso Lora, A., Pérez García, H., Jove Pérez, E., Calvo Rolle, J.L., Martínez de Pisón, F.J., García Bringas, P., Martínez Álvarez, F., Herrero, Á., Fosci, P., Sérgio Filipe, R. (eds.) Hybrid Artificial Intelligent Systems. pp. 354–365. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-08462-0_28
3. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-Rank Adaptation of Large Language Models (2021). https://doi.org/10.48550/arXiv.2106.09685, http://arxiv.org/abs/2106.09685
4. Li, Z., Cao, Y., Xu, X., Jiang, J., Liu, X., Teo, Y.S., Lin, S.W., Liu, Y.: LLMs for Relational Reasoning: How Far are We? In: Proceedings of the 1st International Workshop on Large Language Models for Code. pp. 119–126. ACM, Lisbon Portugal (Apr 2024). https://doi.org/10.1145/3643795.3648387, https://dl.acm.org/doi/10.1145/3643795.3648387
5. Sanz, R., Aguado, E.: Reflective Understanding for Dependable Robots. In: Secchi, C., Marconi, L. (eds.) European Robotics Forum 2024. pp. 216–219. Springer Nature Switzerland (2024). https://doi.org/10.1007/978-3-031-76428-8_41

Deterministic Control and Confined LLM Reasoning in Task-Oriented Dialogue for Service Robots

Álvaro Quintana-Camacho, Juan Diego Peña-Narvaez, José Miguel Guerrero, Francisco Martín-Rico, and Rodrigo Pérez-Rodríguez

Universidad Rey Juan Carlos, Fuenlabrada, Madrid, Spain
aquintana.camacho@proton.me
{juan.pena,josemiguel.guerrero,francisco.rico,rodrigo.perez}@urjc.es

Abstract. Rule-based dialogue systems behave predictably but break outside their scripted domain, while agents driven entirely by a large language model adapt to any domain but cannot guarantee that the conversation reaches its goal. This paper presents a dialogue architecture for service robots that confines the language model to four bounded comprehension operations, extracting slot values, classifying the user intent, selecting the interaction strategy and judging quiz answers, while a deterministic finite-state machine governs the flow and decides when the task is complete. A typed validator keeps hallucinated or malformed model output out of the dialogue state, and the separation makes every delegated operation measurable in isolation. Five benchmarks with 166 manually annotated cases in nine domains compare a cloud model and a local one. Gemini 2.5 Flash reaches an F1 of 0.956 in slot extraction and completes every end-to-end scenario, llama3.2 reveals which operations demand model scale, and neither produces a single false positive when judging quiz answers, a property that derives from the prompt design rather than from the model. The system runs as decoupled ROS 2 packages and is released as open source.

Keywords: Task-oriented dialogue · Large language models · Human-robot interaction · ROS 2 · Service robots.

1 Introduction

A service robot that takes an order, registers a patient or explains a procedure must know which information it still needs, when to insist after an ambiguous answer and when the task is complete. Goal-oriented dialogue differs from open-ended chat in one decisive respect. Either the robot obtains the information the task requires or it does not, and it is always possible to check which of the two happened [8].

The established approaches fail that requirement in complementary ways. Handcrafted dialogue rules behave predictably, but anything outside the anticipated script breaks them [15]. Driving the conversation entirely with a large

language model (LLM) removes that rigidity, but nothing guarantees that the exchange progresses towards a concrete outcome [9]. A deployed robot tolerates neither failure mode, so this work reallocates responsibilities, leaving flow control, state and termination to deterministic logic and invoking the LLM for the linguistic operations that code cannot solve. The model proposes, a typed validator filters and a finite-state machine decides, so the worst a faulty inference can cause is a discarded extraction in a single turn.

The resulting architecture combines three properties that previous systems offered only separately. It adapts to a new domain without retraining, drives every conversation to a verifiable termination and moves across robotic platforms untouched, because all integration happens through ROS 2 interfaces [10]. The evaluation method derives from the same principle. A model that never controls the flow can be benchmarked operation by operation, and running the unmodified engine without ROS 2 or audio hardware verifies that the claimed decoupling exists. We release the five benchmarks and the implementation in the open DialoStack repository[1].

2 Related Work

Task-oriented dialogue has revolved around the frame model since GUS introduced slots to be filled through conversation [2, 8]. Rule-based descendants reached industrial scale with VoiceXML [15] at the price of rigidity, and transfer approaches such as TOD-BERT [16] widened that bottleneck while still requiring labelled data per domain. In-context approaches removed the retraining step by tracking dialogue state from structured prompts alone [6], but they guarantee neither progression to the goal nor an integration path onto a robot. That fragility has since motivated controllers around the model. NeMo Guardrails defines programmable rails without a state machine or termination guarantees [11], Rasa CALM has the LLM translate the conversation into commands executed by deterministic business logic [3] and CoDial compiles a task graph into guardrailing code [12]. Their controllers, however, are compiled, derived or learned rather than designed for inspection, and none defines verifiable termination conditions.

On robots, LLMs first served physical action planning, as in SayCan, which assumes the goal of the interaction is already known [1]. Conversational integrations are closer. Stark et al. hand the whole dialogue and behaviour of a service robot to GPT-4 in Dobby [14], the delegation this work avoids. Grassi et al. deploy their CAIR framework on Pepper and NAO, guiding generation with a knowledge base but controlling the flow through probabilistic transition matrices [5]. No reviewed system combines domain adaptation without retraining, verifiable termination and platform portability.

[1] `https://github.com/aquintan4/DialoStack`

3 Architecture

3.1 Design Principles

The architecture follows one organising decision, the language model contributes reasoning but never control. It is invoked for bounded linguistic operations, receives exactly the context each one needs and must return structured output, and every value it proposes passes a typed validator that discards hallucinated or malformed responses [7]. The same prompt and schema mechanism serves all four operations, slot extraction, intent classification, strategy selection and quiz judgement, so a new operation only adds a template and a schema, not new inference code. Comprehension and generation also stay apart. Understanding requires typed output while replying requires natural text, and serving both from one history makes them interfere, so each dialogue opens two independent model sessions, which also allows retrying a malformed comprehension result invisibly to the user.

Control resides in a small deterministic automaton that drives the turn loop, the timeouts and the cancellation paths and knows nothing about any concrete task. Task logic lives in an interchangeable strategy consulted each turn, so the flow of every conversation is identical and verifiable, and a new interaction mode is one new strategy. Auxiliary subsystems such as emotion detection, gestures and visual feedback add context or expressivity but none is required to complete a task, so when one fails the dialogue core continues in degraded mode, and the system serves one dialogue at a time, since a robot has one voice channel.

Figure 1 shows how these decisions materialise, with the engine reaching the models only through a prompt builder and a generic client. Speech enters through automatic speech recognition (ASR) gated by voice activity detection (VAD) and leaves through text-to-speech synthesis (TTS), both behind an audio abstraction, while the LLM manager and the robot-side integrations stay outside the engine.

3.2 Dialogue Frame and Context

The engine represents the task as a dialogue frame in the tradition of GUS [2], a set of named slots the conversation must fill. The schema arrives with the task or is derived from its natural language description. Each slot declares one of five primitive types and optionally a closed set of canonical values. Scalars replace their value on each update while lists accumulate items across turns. A slot may also be conditional on another taking a concrete value, entering scope only while the condition holds. Figure 2 shows the frame of a medical appointment task in the format the engine consumes, with a conditional slot and a list slot.

Deciding what to ask next only requires the slots still empty, whereas extraction considers every slot in scope, including the filled ones, so a correction over an already registered value does not pass unnoticed. Alongside the frame, a context injected selectively into prompts holds the reference data of the domain, such as a menu or a question bank, and the emotional state of the user perceived through vision, the latter only while the detection is confident and recent.

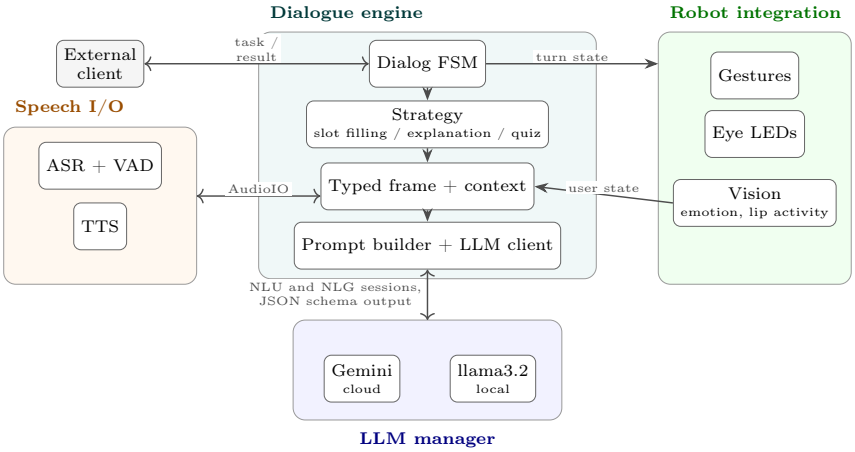


Fig. 1. Overview of the architecture. The dialogue engine depends only on an audio abstraction and an LLM client.

```
{"slots": [
  {"name": "patient_name", "type": "str"},
  {"name": "visit_type",   "type": "str",
   "canonical_values": ["general practice",
       "specialist", "emergency"]},
  {"name": "specialty",    "type": "str",
   "canonical_values": ["cardiology",
       "dermatology", "traumatology"],
   "condition_slot":  "visit_type",
   "condition_value": "specialist"},
  {"name": "symptoms", "type": "list_str"}]}
```

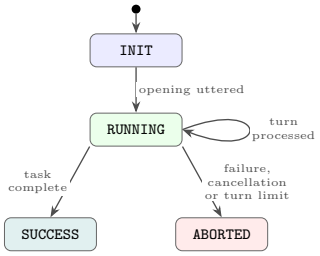


Fig. 2. An example dialogue frame (left) and the control automaton (right). The slot `specialty` exists only while `visit_type` holds `specialist`.

3.3 Control Automaton and Interaction Strategies

Conversational control is an explicit finite-state machine, the four-state automaton on the right of Fig. 2. A dialogue is initialised in `INIT`, moves to `RUNNING` once the opening utterance is spoken and iterates there until it ends in `SUCCESS` or `ABORTED`. Each iteration listens, lets the active strategy process the turn and utters the reply returned with a completion flag. The automaton owns the global limits, tolerating a configurable number of consecutive listening timeouts and bounding the total number of turns.

Three strategies implement the interaction modes behind a common interface. Slot filling extracts in a single pass the new values, the corrections and the list operations of each turn, asks for what is still missing and, once the frame is complete, summarises it and requests an explicit confirmation, so the result is validated by the user rather than inferred. Explanation presents a content and verifies comprehension, reformulating from a different angle a bounded number of times. Quiz walks the user through a question bank and judges each answer

by meaning rather than word by word, a mode aimed at settings such as staff training or cognitive stimulation with therapeutic robots.

A medical appointment over the frame of Fig. 2 illustrates the cycle. The user gives a name and asks for a specialist in one sentence, so extraction fills two slots at once and `visit_type` brings `specialty` into scope, which becomes the next question. The user then answers cardiology but also amends the name, and the same extraction pass registers both changes because filled slots remain visible. With the frame complete the strategy summarises the data, the user confirms and the automaton terminates in `SUCCESS`.

The strategies share the treatment of spoken cancellation, where words such as "no" or "stop" also appear in legitimate answers, so the system never aborts on a first signal. A regular-expression fast path catches unambiguous requests, the model is consulted for ambiguous ones and in either case the user must confirm the abandonment, otherwise the conversation resumes where it was. While the robot speaks its transcriber is muted, and barge-in uses a deliberately conservative signal, because a missed interruption costs a repetition while a false one cuts the robot off mid-sentence [13].

4 Experimental Evaluation

The architecture is implemented as decoupled ROS 2 packages. The dialogue manager exposes the engine through an action that receives the task description, mode, domain and resources and returns the final frame. Speech input combines faster-whisper with Silero VAD, synthesis streams through Piper and a vision package contributes facial emotion and lip activity from MediaPipe. A dedicated node serves inference for Gemini 2.5 Flash in the cloud and llama3.2 under Ollama [4], both forced to schema-constrained JSON output for every comprehension call. As a proof of integration the system was deployed inside a behaviour architecture on a NAO humanoid, gaining gestures, turn-signalling eye LEDs and lip-activity detection without touching the engine.

The engine delegates exactly four operations to the model, slot extraction, intent classification at confirmation, strategy selection from the task description and quiz judgement, and the evaluation package measures each one in isolation by importing the engine modules directly and injecting user turns as text. Five hand-written datasets cover the four operations and an end-to-end simulation, 166 cases over nine domains[2]. Each dataset mixes the habitual conditions of its operation with the edge cases that break it. Extraction combines simple and multi-slot requests with corrections and turns where nothing should be extracted, intent classification balances twelve to fourteen samples per class including irony, incomplete sentences and marked indecision, task-mode selection labels 25 short descriptions split between the two modes, quiz judgement pairs 21 correct answers under paraphrase and spelling variants with 15 plausible wrong ones, and

[2] Ice cream and pizza ordering, hotel booking, car rental, medical appointments, restaurant reservation, café service, general knowledge quizzes and employee training.

Table 1. Headline results of the four component benchmarks, with the number of cases in parentheses.

Benchmark	Metric	Gemini 2.5 Flash	llama3.2
Slot extraction (38)	P/R/F1	0.915/1.000/**0.956**	0.435/0.698/**0.536**
Intent classification (50)	Acc./macro F1	0.900/0.900	0.580/0.576
Task mode (25)	Accuracy	1.000	0.800
Quiz judgement (36)	Acc./FP	0.972/0	0.778/0

Table 2. Confusion matrices for intent classification, with rows as true classes and each cell showing Gemini 2.5 Flash / llama3.2 counts.

True \ predicted	*confirms*	*corrects*	*question*	*unclear*
confirms	**14 / 14**	0 / 0	0 / 0	0 / 0
corrects	0 / 7	**12 / 4**	0 / 0	0 / 1
question	1 / 6	1 / 0	**10 / 6**	0 / 0
unclear	3 / 7	0 / 0	0 / 0	**9 / 5**

the simulation adds two dialogues designed to end in cancellation. With 17 to 50 cases per benchmark, the goal is qualitative coverage rather than statistical power, so the reported figures characterise error patterns rather than tight estimates.

The metrics follow the cost structure of each operation. Extraction reports precision, recall and F1 because one turn may yield several values and inventing one is a different error from missing one. Intent classification reports accuracy with macro F1 and the full confusion matrices, and quiz judgement counts false positives separately. The two models represent the intended deployment scenarios, cloud and local inference, and the prompts were developed with Gemini as the working model, an origin that should be kept in mind when reading the llama3.2 figures.

4.1 Component Benchmarks

Table 1 gathers the headline figures. **Slot extraction** is the most frequent and most consequential operation, since a wrong value corrupts the frame and can reach confirmation. Over 43 expected extractions in 38 cases, Gemini reaches perfect recall in every type and domain, and its four false positives share one pattern, committing a canonical value when the user mentions an option without choosing it. llama3.2 fails differently per type. On scalar strings recall stays at 0.947 while precision collapses to 0.400, since the model infers values the user never gave. On lists the opposite happens, with perfect precision and a recall of 0.214 that means most enumerated items are dropped.

Intent classification decides whether the user confirms the summarised data, corrects a value, asks a question or answers ambiguously. Table 2 shows both confusion matrices. Four of the five errors of Gemini involve genuinely ambiguous replies, and all twelve corrections are classified without error, so the system never closes a task while the user is amending a value. llama3.2 shows a

structural bias towards confirmation instead. It recognises the 14 real confirmations but drags 20 of the 36 remaining replies into that class, wrongly closing six of every ten tasks in which the user was correcting, asking or hesitating. The architecture cannot absorb this error, because nothing downstream can reveal that a confirmation was in fact a correction.

Task mode classification happens once, before the first utterance, and a mistake makes the robot interact in the wrong mode from its first sentence. Gemini classifies the 25 descriptions without error, while the five errors of llama3.2 all label explanation tasks as slot filling, leaving explanation recall at 0.583. **Quiz judgement** is the operation most sensitive to false positives, since accepting a wrong answer closes the question with an error the user never learns about. Neither model produces a single false positive over the 15 plausible wrong answers, including years off by one. The conservative prompt rejects when in doubt, and the remaining errors are false negatives that force the user to rephrase, one abbreviation for Gemini and eight semantically equivalent reformulations for llama3.2.

4.2 End-to-End Simulation and Latency

The simulation runs full dialogues through the real engine against a scripted user. Gemini completes all 15 scenarios that should succeed, in 3.4 turns on average, with an outcome accuracy of 1.000 and a frame accuracy of 0.909, its only blemishes two fields holding a non-canonical variant of the correct value. Both models close the two abandonment scenarios with a clean cancellation, so the detection mechanism is model-independent. llama3.2 completes 14 of the 15, failing one car rental where extraction errors accumulate until the frame cannot be filled, and its frame accuracy of 0.530 quantifies how component errors compound over a dialogue.

Across the 149 calls of the component benchmarks, mean latency per operation ranged from 0.67 s for quiz judgement to 0.87 s for extraction. A traced restaurant dialogue on the full stack kept the language computation of a turn near one second, while its 68.5 s of total wall time were dominated by the listening window and the speech synthesis, which makes the model anything but the bottleneck of the interaction.

5 Discussion and Conclusions

The benchmarks separate what the architecture protects from what depends on the model. Slot extraction is the most demanding operation linguistically, and the gap between models there explains most of the deterioration in frame accuracy during simulation. Quiz judgement shows the opposite pattern, since its zero false positives travel with the prompt rather than with the model. Intent classification sits in between, and the confirmation bias of llama3.2 makes model choice part of the reliability guarantees of the system. The local scenario still

matters, since on-edge inference sends no data out and works without connectivity, which in privacy-sensitive deployments may outweigh the accuracy gap. With the method established, the natural push is towards larger quantised local models with prompts developed specifically for them. The main limitation is that the simulated user cannot reproduce the oral variability of real users, so a user study and the finer turn-taking that turn latency demands are the priority lines for the interaction experience.

Confining the language model to bounded linguistic operations and leaving flow, state and termination to a small deterministic automaton thus yields domain adaptation without retraining, verifiable termination and platform portability at once. The engine has been integrated unmodified into a behaviour architecture running on a real humanoid, and since every platform dependency passes through ROS 2 interfaces and an audio abstraction, carrying it to other robots is direct.

Acknowledgments. This work originated as a Bachelor's Thesis at URJC.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Ahn, M., Brohan, A., Brown, N., et al.: Do as I can, not as I say: Grounding language in robotic affordances. arXiv:2204.01691 (2022)
2. Bobrow, D.G., Kaplan, R.M., Kay, M., Norman, D.A., Thompson, H., Winograd, T.: GUS, a frame-driven dialog system. Artificial Intelligence **8**(2), 155–173 (1977)
3. Bocklisch, T., Werkmeister, T., Varshneya, D., Nichol, A.: Task-oriented dialogue with in-context learning. arXiv:2402.12234 (2024)
4. González-Santamarta, M.Á., Rodríguez-Lera, F.J., Sobrín-Hidalgo, D., et al.: Integrating quantized LLMs into robotics systems as edge AI to leverage their natural language processing capabilities. arXiv:2506.09581 (2025)
5. Grassi, L., Recchiuto, C.T., Sgorbissa, A.: Enhancing LLM-based human-robot interaction with nuances for diversity awareness. In: Proc. IEEE RO-MAN. pp. 2287–2294 (2024)
6. Hudeček, V., Dušek, O.: Are large language models all you need for task-oriented dialogue? In: Proc. SIGDIAL 2023. pp. 216–228 (2023)
7. Ji, Z., Lee, N., Frieske, R., et al.: Survey of hallucination in natural language generation. ACM Computing Surveys **55**(12), 248:1–248:38 (2023)
8. Jurafsky, D., Martin, J.H.: Speech and language processing. 3rd edn. (online draft), chap. K: Task-Oriented Dialogue (2024)
9. Kambhampati, S.: Can large language models reason and plan? Annals of the New York Academy of Sciences **1534**(1), 15–18 (2024)
10. Macenski, S., Foote, T., Gerkey, B., et al.: Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022)
11. Rebedea, T., Dinu, R., Sreedhar, M., Parisien, C., Cohen, J.: NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In: Proc. EMNLP 2023 (System Demonstrations). pp. 431–445 (2023)
12. Shayanfar, R., Luo, C.F., et al.: CoDial: Interpretable task-oriented dialogue systems through dialogue flow alignment. arXiv:2506.02264 (2025)

13. Skantze, G.: Turn-taking in conversational systems and human-robot interaction: A review. Computer Speech & Language **67**, 101178 (2021)
14. Stark, C., Chun, B., et al.: Dobby: A conversational service robot driven by GPT-4. In: Proc. IEEE RO-MAN. pp. 1362–1369 (2024)
15. W3C: VoiceXML version 2.0. W3C Recommendation (2004)
16. Wu, C.S., Hoi, S.C.H., Socher, R., Xiong, C.: TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In: Proc. EMNLP 2020. pp. 917–929 (2020)

An Autonomous Attention System for Social Robots Based on Multimodal Reasoning and Behavior Trees

Sergio Cobos Blanco, Juan Diego Peña-Narváez, José Miguel Guerrero, Francisco Martín, and Rodrigo Pérez-Rodríguez

Universidad Rey Juan Carlos, Escuela de Ingeniería de Fuenlabrada, Fuenlabrada, Spain
sergiocblanco2004@gmail.com
{juan.pena,josemiguel.guerrero,francisco.rico,rodrigo.perez}@urjc.es

Abstract. Social robots operating in shared and dynamic environments must actively manage what they attend to in order to keep task-relevant information available during execution. This requires more than processing sensory data: the robot must decide which elements of the scene are relevant, how its perceptual and actuation resources should be used, and how to react when the attended information becomes unreliable or temporarily unavailable. This paper presents an autonomous attention system for social robots that can be integrated as an independent and reusable subsystem within a broader robotic architecture. Given the task context, the current robot activity, and the available actuation capabilities, the system selects, prepares, executes, and supervises an attentional strategy aimed at maintaining access to relevant information. The approach combines contextual reasoning with structured execution, while keeping the attention logic separate from the specific perception and control components used in a particular platform. The system was evaluated in a real robot setup through attentional strategy selection, execution preparation, failure notification, and visual tracking experiments. The results show that the proposed approach can autonomously manage attention requests, support high-level attentional decisions, and maintain an attentional focus during robot operation, while identifying reasoning latency and perceptual stability as the main factors affecting performance. The project's repository is available at https://github.com/geriabot/attention_system.

Keywords: Robotic attention · Social robots · Contextual reasoning · Attentional behaviors · Modular robotic architectures

1 Introduction

Social robots operate in environments where the information required for a task is not always directly available to their sensors. The usefulness of perception depends not only on the detection algorithm, but also on the robot's pose, sensor

orientation, visual coverage, and spatial relationship with relevant objects or people. Therefore, perception in autonomous robots is not purely passive: the robot must decide how to configure its body, sensors, and auxiliary modules to capture useful information during execution.

This paper addresses autonomous attention for social robots. We understand attention as an active coordination mechanism between perception and action, not as a post-processing filter over acquired data. Attending may involve orienting the robot, keeping an object inside the camera field of view, activating a detector, tracking a transform, or selecting a recovery behavior when the focus is lost.

The proposed system allows an external mission layer to request attention by providing task context. From this request, the attention subsystem decides how attention should be managed, infers any missing execution information, coordinates the required perception and actuation resources, and reports its status.

The contributions are: (i) a modular architecture that treats attention as an autonomous subsystem; (ii) a reasoning pipeline that uses a multimodal large language model (MLLM) for behavior selection and parameter inference; (iii) behavior-tree-based execution with explicit recovery and controlled deactivation of external components; and (iv) an experimental validation on TurtleBot 2 with open-vocabulary perception and transform-based tracking.

2 Related Work

Robotic attention is a key capability for socially interactive robots. Ferreira and Dias [1] review attentional mechanisms for social robots, emphasizing the selection and prioritization of relevant stimuli. Early work on Kismet showed that visual attention is closely coupled with behavior, as gaze orientation depends on perceptual, motivational, and social constraints [2]. This perspective is also related to active perception, in which agents act or reconfigure their sensing resources to obtain task-relevant information [3]. Lanillos et al. [4] further proposed an artificial attention system embedded in a perception–action loop.

Other works have addressed the architectural integration and control of attention. Martín et al. [5] manage visual attention through a client–server approach within a cognitive architecture, demonstrating the relevance of decoupling focus selection and attentional control from other robotic functions. Component-based architectures similarly support reuse and adaptability [6], while behavior trees provide a modular and reactive execution model [7]. Large models have also recently been applied to contextual reasoning in social and assistive robotics [8, 9].

However, prior work mainly addresses isolated aspects of robotic attention. This leaves a clear research gap: their integration into a reusable, task-driven autonomous subsystem that, once invoked by a higher-level architecture, selects attentional strategies, coordinates perception and actuation, supervises execution, and recovers from critical conditions. Our approach fills this need by combin-

ing MLLM-based high-level decision making with behavior-tree-based reactive execution.

3 Attention System Design

The proposed system is conceived as an autonomous attention subsystem that can be integrated into a broader robotic architecture. At the robot level, attention is treated as a specialized subsystem alongside perception, navigation, HRI, or other functional modules. Internally, however, the attention subsystem is organized as an attention system composed of several components responsible for request handling, contextual reasoning, behavior execution, supervision, and communication with external resources. Its internal organization and execution logic are independent of the host architecture, allowing the subsystem to be integrated into different robotic platforms through external interfaces and, when necessary, platform-specific configuration parameters.

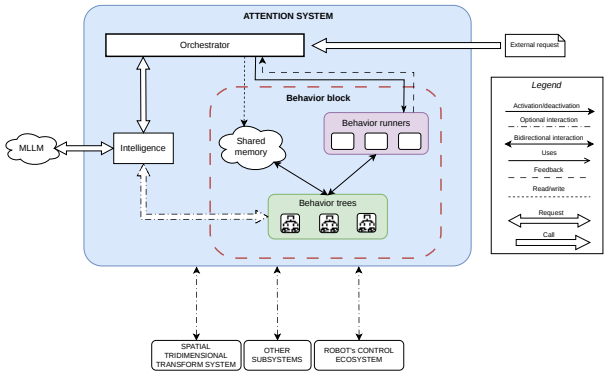


Fig. 1. Architecture of the autonomous attention system.

Figure 1 shows how these components interact with the external client and with the perception, spatial transformation, and robot-control subsystems. The *orchestrator* coordinates the lifecycle of each attention request: it receives the task context, filters the available behaviors according to the actuation capabilities provided in the request, stops any previous attentional behavior if necessary, and activates the selected behavior runner. During execution, it monitors the state reported by the runner and notifies success, failure, or interruption to the external architecture.

The *intelligence* component performs the reasoning operations required by the system. It selects the most suitable attentional behavior among the compatible candidates and may also infer missing execution parameters, such as the target class or the identifier of the detection that should become the attentional target. The model receives a structured request built from the task description, the current robot activity, the contextual information, and the available behaviors or required inputs. Its response is interpreted as structured output, allowing the selected behavior or inferred parameters to be used automatically

by the rest of the system. For example, behavior selection can be represented by a minimal output containing only the identifier of the selected behavior, such as `"attention_action_id": 1`.

The *behavior block* contains the behavior runners, the executable behavior definitions, and the shared memory used during execution. A *behavior runner* is the execution manager associated with a selected behavior tree: it starts the tree, ticks it during execution, handles interruptions, and reports its state to the orchestrator. Each attentional behavior is represented as a behavior tree, which combines perception requests, actuation commands, condition checks, reasoning calls, and recovery logic within the same execution structure. The *shared memory* allows the orchestrator and behavior nodes to exchange the task context, known parameters, pending values, available capabilities, and inferred information without direct coupling.

When an attention request is received, the orchestrator stops any active behavior, asks the intelligence component to select a compatible one, writes the available execution data to shared memory, and activates the corresponding runner. It then supervises execution and handles completion, failure, or interruption.

3.1 Attentional behaviors

In the proposed system, three attentional behaviors instantiate the general behavior-based formulation. They follow a common execution pattern: missing parameters are inferred when required, the necessary perception resources are activated, a target spatial reference is generated, and the corresponding tracking component is started. The suffix *Rot* denotes behaviors based on base rotation, whereas *Art* denotes behaviors that rely on an articulated element of the robot.

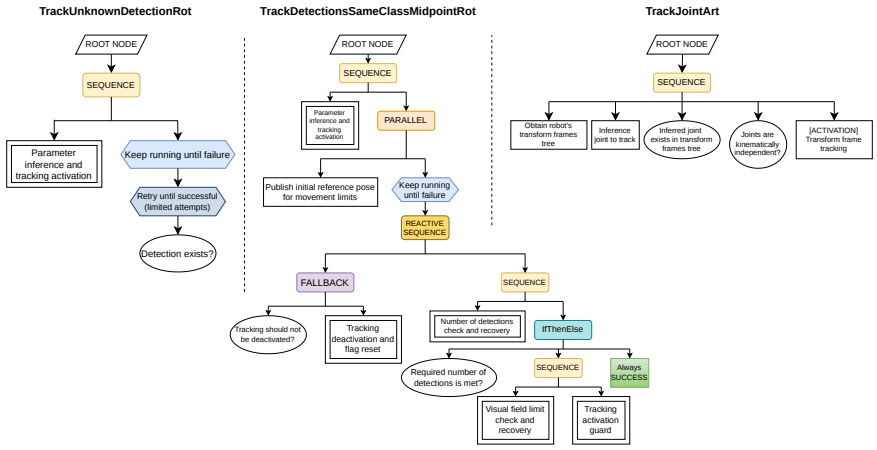


Fig. 2. The three designed attentional behavior trees.

TrackUnknownDetectionRot is used when the robot must keep a visual element in view but neither the target class nor the specific detection identifier

are known at activation time. The behavior infers the target class from the task context, activates the corresponding visual detector, selects a concrete detection once available, generates its spatial target reference, and starts rotational tracking. Temporary detection losses are handled through a bounded number of recovery attempts. If detection is restored within this limit, the tree continues running; otherwise, it reports failure.

TrackDetectionsSameClassMidpointRot addresses cases in which the attentional focus is a group of detections from the same class rather than a single object. The behavior first infers the detection class to be tracked and, once the midpoint target reference has been generated and tracking is active, supervises whether the conditions for normal tracking remain valid. If some detections are available but their number is lower than required, tracking can be temporarily deactivated and the robot can scan the scene to recover the missing detections. If detections appear near both lateral limits, the robot can move backwards to increase the covered field of view. Conversely, if the detections are sufficiently clustered, the robot can move forwards to reduce the covered field of view. Once the required perception conditions are restored, tracking is activated again.

TrackJointArt is intended for situations in which the robot must maintain visual access to one of its own joints during an action. The behavior obtains a compact representation of the robot kinematic structure, infers the joint to observe, checks that the corresponding spatial reference exists, and verifies that the joint used to orient the sensor is kinematically independent from the target joint. This prevents circular dependencies in which moving the sensor also moves the target being observed. If these conditions hold, tracking of the selected joint reference is activated.

Together, these behaviors show that attentional behaviors are structured procedures that combine reasoning, perception activation, target generation, actuation, supervision, recovery, and controlled deactivation of external components. Therefore, new attentional behaviors can be added by defining new behavior structures and their associated parameters, without changing the general organization of the attention system.

4 Implementation

During execution, a global status variable coordinates the active attention request and reports whether it is running, has succeeded, has failed, or has been interrupted. The MLLM is queried only at discrete points to select the attentional behavior or infer missing inputs, such as the object class to detect or the detection identifier to track. In the implemented system, this reasoning can be performed either by local models, such as Qwen3VL-8B-Instruct, or by a remote Gemini-based backend. This keeps continuous control outside the reasoning loop while allowing different model backends to be used for high-level attentional decisions.

The attentional behaviors are implemented with BehaviorTree.CPP [10]. Its blackboard pattern stores execution data and shared resources used by the

BT nodes. Some behaviors activate external components, such as the detector, the target-reference generator, or the tracking controller; these components are stopped by dedicated deactivation nodes when the tree succeeds, fails, or is interrupted.

The visual pipeline uses OMDet Turbo [11] as an open-vocabulary detector, since the target class can be provided as a textual prompt inferred by the attention system. The camera image is processed to obtain 2D detections, which are converted into spatial references using RGB-D depth when available or camera intrinsics otherwise. The system exposes either a selected detection or the midpoint between several detections as the target reference. On TurtleBot 2, the controller converts the horizontal angular error to this reference into angular velocity commands for base rotation; when visual coverage must be adjusted, linear velocity commands move the robot forward or backward.

5 Evaluation

The evaluation verifies the complete execution flow of the attention system on a real robotic platform. The goal of the evaluation is not to assess the perception or actuation components in isolation, but to determine whether the attention system can use them correctly within the complete execution flow.

The experiments were carried out in an indoor laboratory environment using a TurtleBot 2 platform equipped with an Orbbec Astra RGB-D camera. The attention system, perception nodes, actuation nodes, and local MLLM backends were executed on a laptop with an Intel Core i7-14700HX CPU, 32 GiB of RAM, and an NVIDIA GeForce RTX 4060 GPU. The software environment was based on Ubuntu 24.04, ROS 2 Jazzy, CUDA, and `llama_ros`.

5.1 Autonomous Behavior Selection

The first experiment evaluates whether the attention system can autonomously select the expected attentional behavior from the task context and the robot activity. For this purpose, three metrics are considered: activation latency (L_{act}), reasoning latency (L_{reason}), and selection accuracy (A_{sel}). The activation latency measures the time, in seconds, between the request sent to the attention system and the response returned by the orchestrator. The reasoning latency measures the time, in seconds, spent by the intelligence pipeline to select the behavior. Finally, the selection accuracy is the percentage of requests in which the selected behavior matches the expected one.

A set of 8 tasks and 17 task-activity combinations was defined. Each combination was repeated 10 times, testing 170 requests per model. Four local MLLMs were evaluated: MiniCPM-V-4.5, Qwen2.5-VL-3B-Instruct, Qwen3VL-8B-Instruct, and SmolVLM2-2.2B-Instruct. Table 1 summarizes the results.

Model	$\overline{L}_{act}$ (s)	$\sigma_{L_{act}}$ (s)	$\overline{L}_{reason}$ (s)	$\sigma_{L_{reason}}$ (s)	A_{sel} (%)
`MiniCPM-V-4.5`	1.557	0.225	1.172	0.226	84.12
`Qwen2.5-VL-3B-Instruct`	1.112	0.237	0.729	0.237	49.41
`Qwen3VL-8B-Instruct`	1.538	0.168	1.154	0.168	83.91
`SmolVLM2-2.2B-Instruct`	0.988	0.006	0.688	0.000	43.53

Table 1. Results of the autonomous attentional behavior selection. Latencies are expressed in seconds.

The results show that the reasoning pipeline accounts for most of the activation latency. In the two models with the highest selection accuracy, MiniCPM-V-4.5 and Qwen3VL-8B-Instruct, the reasoning stage takes approximately 1.17 s and 1.15 s, respectively, within a total activation latency close to 1.5 s. This indicates that the temporal cost added by the rest of the system is low compared with the MLLM inference time. However, this latency is acceptable during the activation phase, since it is not part of the continuous tracking loop and allows the system to obtain a reliable selection of the attentional behavior.

5.2 Initial Behavior Preparation

The second experiment evaluates the time required to prepare the selected attentional behavior before tracking is activated for the first time. This preparation latency (L_{prep}) is measured in seconds from the first tick of the behavior execution to the first successful activation of tracking. This metric is relevant because attentional behaviors may require reasoning, external perception requests, target generation, and condition checks before they can start acting on the robot.

The experiment uses the same task set defined for behavior selection. In each execution, the system records the first tick of the selected behavior and the instant at which tracking is activated for the first time. The experiment focuses on the two models with the best behavior-selection performance, MiniCPM-V-4.5 and Qwen3VL-8B-Instruct. For each combination of model and attentional behavior, 50 executions were performed. Table 2 reports the preparation latency for the implemented attentional behaviors.

Model	Behavior	$\overline{L}_{prep}$ (s)	$\sigma_{L_{prep}}$ (s)
`MiniCPM-V-4.5`	`TrackDetectionsSameClassMidpointRot`	5.658	0.744
`MiniCPM-V-4.5`	`TrackJointArt`	3.821	0.217
`MiniCPM-V-4.5`	`TrackUnknownDetectionRot`	12.595	0.765
`Qwen3VL-8B-Instruct`	`TrackDetectionsSameClassMidpointRot`	3.688	0.363
`Qwen3VL-8B-Instruct`	`TrackJointArt`	2.941	0.229
`Qwen3VL-8B-Instruct`	`TrackUnknownDetectionRot`	7.970	0.706

Table 2. Initial behavior preparation latency. Latencies are expressed in seconds.

The preparation latency depends both on the model and on the internal structure of each behavior. *TrackJointArt* obtains the lowest preparation times and small standard deviations, which indicates a more stable preparation phase. In contrast, *TrackUnknownDetectionRot* requires a longer preparation time because it includes more steps before tracking can start, including the inference of the object class and the specific detection identifier. Qwen3VL-8B-Instruct

reduces the preparation latency with respect to MiniCPM-V-4.5 in all three behaviors, while maintaining moderate standard deviations. These latencies do not affect the continuous tracking loop, but they determine the initial delay before the attentional behavior starts acting on the robot.

5.3 Failure Notification

The third experiment evaluates whether behavior failures are reported to the external architecture with low delay. The failure-notification latency, (L_{fail}), is measured as the time between the behavior execution reporting failure and the publication of that failure in the global attention-system status. This was evaluated over 170 executions, obtaining a mean latency of (0.168 s) with a standard deviation of (0.021 s).

This result indicates that failure reporting introduces a low and stable delay, allowing the external component that requested attention to react quickly when the active behavior stops executing correctly.

5.4 Recovery Mechanisms

The recovery mechanisms of *TrackDetectionsSameClassMidpointRot* were evaluated by forcing situations in which the behavior had to leave normal tracking and recover the attentional focus. These situations include the temporary loss of the conditions required for normal tracking, the need to approach grouped detections, and the presence of tracked detections close to both lateral limits of the visual field. For each recovery case, 15 executions were performed. During each execution, the system recorded the delays associated with the start of recovery, the deactivation of tracking, and the return to normal tracking after the recovery condition disappeared.

Measured delay	$\overline{L}$ (s)	σ_L (s)
Recovery start	0.063686	0.336691
Tracking deactivation	0.219412	0.089488
Recovery exit	0.390173	0.909402

Table 3. Recovery latency results for *TrackDetectionsSameClassMidpointRot*. Latencies are expressed in seconds.

Table 3 shows that the behavior reacts quickly when tracking can no longer continue under valid perception conditions. Recovery starts in less than (0.1 s) on average, tracking is deactivated after approximately (0.22 s), and normal execution resumes in less than (0.4 s), although with higher variability in this last stage. Overall, these results indicate that the behavior can trigger and complete recovery procedures within short time intervals, allowing it to react effectively to changes in perception conditions.

5.5 TurtleBot 2 Attentional Tracking

The final experiment evaluates the tracking behavior on TurtleBot 2 using the transform-tracking controller described in Section 4. The controller rotates the

mobile base toward a target spatial reference. The recorded trace was grouped into three motion ranges according to the median speed of the target. This grouping avoids treating each segment as an independent experiment and reduces the effect of outliers caused by occasional jumps in the perception output.

The reported metrics are the mean absolute yaw error (MAE_{yaw}), expressed in radians; the percentage of saturated angular velocity commands (S_{yaw}), expressed as a percentage; and the estimated target loss, also expressed as a percentage. The median target speed is expressed in m/s. Table 4 reports the tracking results.

Motion range	Median target speed (m/s)	MAE_{yaw} (rad)	S_{yaw} (%)	Target loss (%)
Low	0.248	0.158	8.738	3.147
Medium	0.731	0.183	5.816	0.871
High	1.548	0.207	12.688	0.699

Table 4. TurtleBot 2 attentional tracking results for different target motion ranges.

The tracking results indicate that the system maintains the target inside the visual field in most samples, with an estimated target loss below (3.2%) in all motion ranges. The percentage of saturated angular velocity commands also remains below (13%), showing that the robot does not need to rely continuously on maximum control commands to preserve the attentional focus. The clearest effect of increasing target speed is observed in the yaw error, which grows from ($MAE_{\text{yaw}} = 0.158$ rad) in the low-motion range to ($MAE_{\text{yaw}} = 0.207$ rad) in the high-motion range. This suggests that the system can keep the target attended under the tested conditions, although tracking precision decreases when target motion becomes more demanding or when perception instabilities introduce jumps in the followed spatial reference.

Although the tracking experiment was conducted with a TurtleBot 2 platform using base rotation, the evaluated attentional strategy is not conceptually limited to this actuation mechanism. On platforms with actuated necks, torsos, or camera joints, the same strategy could be instantiated by replacing the tracking component, as long as the platform is able to orient its perceptual resources toward the selected spatial reference.

6 Conclusions and Future Work

This paper presented an autonomous attention system for social robots, designed as an independent subsystem that can be requested by a higher-level robotic architecture during task execution. The system selects an attentional behavior from contextual information, prepares the required inputs, executes the corresponding behavior tree, and supervises its execution until completion, failure, or interruption.

The implementation shows that the proposed architecture can combine MLLM-based contextual reasoning with reactive behavior execution. The experimental results indicate that the reasoning pipeline is the main contributor to the activation latency, while the rest of the system introduces a smaller overhead. The best-performing evaluated models achieved behavior-selection

accuracies above 80%, with activation latencies around 1.5 s. The implemented behaviors were also able to prepare their execution, report failures with low latency, and maintain visual attention on a target using a TurtleBot 2 platform. These results suggest that the approach is suitable for high-level attentional decisions that do not require hard real-time control.

The evaluation is limited to three attentional behaviors, one platform, controlled indoor scenarios, visual perception, and internal metrics; performance also depends on the selected MLLM and external perception reliability. Generalization beyond these settings remains to be validated.

Future work will extend the behaviors, platforms, sensing modalities, and interaction scenarios; compare the proposal with other robotic attention systems; analyze reasoning failures; incorporate user- and task-oriented metrics; and improve robustness to unreliable perception.

References

1. João Filipe Ferreira and Jorge Dias. Attentional mechanisms for socially interactive robots: A survey. *IEEE Transactions on Autonomous Mental Development*, 6(2):110–125, 2014.
2. Cynthia Breazeal, Aaron Edsinger, Paul Fitzpatrick, and Brian Scassellati. Active vision for sociable robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 31(5):443–453, 2001.
3. Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.
4. Pablo Lanillos, João Filipe Ferreira, and Jorge Dias. Designing an artificial attention system for social robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4171–4178, 2015.
5. Francisco Martín, José Ginés, Francisco J. Rodríguez-Lera, Ángel M. Guerrero-Higueras, and Vicente Matellán. Client-server approach for managing visual attention, integrated in a cognitive architecture for a social robot. *Frontiers in Neurorobotics*, 15:630386, 2021.
6. Alex Brooks, Tobias Kaupp, Alexei Makarenko, Stefan Williams, and Anders Oreback. Towards component-based robotics. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 163–168, 2005.
7. Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, 2018.
8. Abraham Casas, Jesús Martínez-Gómez, and Luis de la Ossa. Efficient real-time scene description with vision-language models. In *Proceedings of the 25th International Workshop on Physical Agents (WAF 2025)*, pages 103–116. Universidad Politécnica de Cartagena, 2025.
9. Antonio Blanco-Castillo, Jackie Lee Chong Ojeda, Alicia Condón, and Pedro Núñez. Enhancing cognitive therapies for older adults with ebo: An autonomous social robot system integrating llms and therapist-in-the-loop. In *Proceedings of the XXIV Workshop of Physical Agents*, pages 31–45. Universidad de Alicante, 2024.
10. BehaviorTree.CPP Contributors. BehaviorTree.CPP, 2026.
11. Tiancheng Zhao, Peng Liu, Xiangyu He, Lu Zhang, and Kyusong Lee. Real-time transformer-based open-vocabulary detection with efficient fusion head, 2024.

Fast Where Safe, Careful Where Risky: Anticipatory Phase Coupling with Per-Segment Precision Learning for Robot Manipulation

Pablo Bustos[1], Jorge Calderón[1], Alejandro Torrejón[1], and Shahriar Hassan[1]

RoboLab, Universidad de Extremadura, Spain
pbustos@unex.es
http://robolab.unex.es

Abstract. A person who keeps doing a familiar task gets better at it without being graded: they grow confident, move faster, and stop double-checking. We give a robot the same kind of self-improvement on a manipulation skill it already knows — with no reward signal and no offline policy search. We treat skill as *model precision* (a confidence, in the active-inference sense) that the robot builds up whenever the task succeeds, lowers when it is surprised, and that quietly reshapes how boldly it acts. We use this precision two ways. First, each stage of the task — approach, insert, lift, place, retreat — earns its own confidence from its own outcome, so the safe stages speed up while the risky ones stay careful. Second, and centrally, each stage becomes *anticipatory*: it prefers to finish in a state the next stage can continue from — minimising the *expected* free energy of what comes next — so the robot never commits to a pose it cannot proceed from. The grasp, for example, is chosen so the bottle can still be lifted from it, turning a chronically failing lift into a reliable one, while the approach hands its leftover alignment to the insertion. On a Kinova Gen3 in simulation this shortens the whole task from 10.1 s to 5.3 s over ten cold-start attempts (−48%, mean of three runs), with reliability preserved. The two effects are complementary: *anticipation clears the way, precision accelerates along it.*

1 Introduction

Robots are increasingly expected to acquire and refine manipulation skills on their own. The dominant paradigm for this is reinforcement learning. It is powerful, but it is data-hungry, depends on careful reward design, and explores in ways that are unsafe on physical hardware. This makes it a poor fit for an agent that already knows, roughly, how to perform a task and simply needs to get *better* at it. We therefore ask a narrower, more practical question. Can an agent that has been specialised by hand on a simple object-mediated affordance, here a pick-and-place, *fine-tune itself* over a handful of executions? We want it to become faster while remaining at least as reliable, purely as a by-product of doing the task, with no reward and no offline policy search.

We approach this through active inference, in which behaviour minimises (expected) free energy. Free energy is a tractable upper bound on *surprise*, the

negative log-evidence of the agent's observations under its own model; we give the concrete expression in Sec. 3.2. A second quantity, *precision*, is the confidence the agent places in that model, and it sets the balance between acting on the model and deferring to sensory evidence [3]. We operationalize skill as exactly this precision. Each time the agent executes the affordance and its predictions are confirmed, its model precision grows. The grown precision then makes the action more model-driven, so the agent reaches more directly, commits sooner, and moves faster. A failure deflates precision, which makes the process self-calibrating: confidence rises only as fast as outcomes keep confirming it. No reward is involved, and improvement is simply the consequence of acting and observing that the model holds.

A single, global confidence occupies only one point on the speed–reliability trade-off. But the segments of a manipulation differ sharply in risk: free-space transport can cruise, while contact-rich insertion and set-down must stay careful. Our first contribution *partitions* the precision per motion segment, so each segment accumulates its own confidence from its own outcome, localising both credit and caution. This sets how fast each segment dares to move, but it leaves a deeper inefficiency. A greedily chosen phase can reach a state from which the *next* phase is impossible. In our case, a grasp can leave the arm in a pose where the straight-up lift is near-singular, twisting and tipping the bottle even though the grasp looked perfect. Our second, and central, contribution makes each phase *anticipatory*: its preference minimises the *expected* free energy of the phase that follows. The grasp commits to the approach azimuth whose manipulability survives the lift, and the approach hands its residual orientation to the insertion. Anticipation clears the near-singular floor, and the *learned* precision then accelerates along the path it opens. We summarise this as *anticipation clears the way, precision accelerates along it*: a phase that ends where the next begins neither stops and restarts nor backs out of a dead end.

We demonstrate the approach on a 7-DoF Kinova Gen3 arm in simulation. In summary, we contribute:

- a formulation of skill acquisition as accumulated model precision from confirmed outcomes — reward-free, offline-search-free, and self-calibrating (Sec. 4);
- a per-segment partition of that precision that localises credit *and* caution — a surprise deflates only the failing segment, not the whole skill (Sec. 4);
- an *anticipatory* use of precision in which each phase's preference minimises the expected free energy of the next — a manipulability-aware grasp-azimuth choice and an approach-to-insertion hand-off — which a component ablation shows clears the dominant obstacle to reliability (and, at near-singular poses, to speed) (Sec. 5); and
- an empirical study — per-segment skill divergence, the lift cured from near-always-failing to 30/30 clean, a $\sim$48% whole-episode speed-up with success preserved, and a component ablation that separates anticipation's geometric gain from precision's learned, self-calibrating one (Sec. 6).

2 Related Work

Active inference casts perception and action as minimisation of (expected) free energy, in which *precision* — the confidence assigned to predictions — plays the role of gain and attention, and behaviour balances pragmatic and epistemic value [3,4]. On robots, active inference has driven manipulators adaptively at the *dynamics* level — a model-free joint-torque law robust to large unmodeled dynamics (link-mass error, payload change) and transferable sim-to-real [11,7]. That robustness is *orthogonal* to ours: we act at the behavioral level and command joint velocities to a servo that absorbs the dynamics, so the two stacks *compose* — a dynamics-AIC beneath our precision-learning behavioral-AIC. Closest to us, active inference has been used to *learn precision* online [1] — there the learned quantity is sensory precision for fault-tolerant low-level control, whereas we accumulate *model* precision from task outcomes to learn *skill*, reward-free, partition it per motion segment, and deploy it anticipatorily. The structure nearest to ours is hierarchical active inference, recently scaled to long-horizon mobile-manipulation benchmarks by pairing high-level discrete skill *selection* with a whole-body active-inference controller [10]. That line *composes and selects* among skills to reach long-horizon goals; we are complementary, taking a single specialised skill and making it measurably *better* — faster and more reliable — by accumulating precision from its own outcomes and, crucially, by letting each phase *anticipate* the next via a one-step expected-free-energy choice at the phase boundary, rather than a softmax policy posterior over a deep tree. Our within-phase controller is a reactive resolved-rate QP in the spirit of NEO [5]; the via-point formulations of movement primitives [6,9] are the nearest sequencing analogue, the difference being that our vias carry *learned precisions* gating soft transitions and that a phase's target anticipates the feasibility of those downstream. The premise of getting faster by repeating a task also recalls iterative learning control [2], which refines a feedforward command to track a *fixed reference* better each trial; we instead accumulate a confidence that reshapes a reactive policy's speed, geometry and tolerances, with no reference trajectory and no per-trial error integral.

3 Problem Setting and the Sequencer

The system has three layers. A *continuous controller* (Sec. 3.3) realises, within any phase, a precision-weighted preference over the tool pose as a resolved-rate joint-velocity command. A *discrete sequencer* (Sec. 3.4) chains the phases of the affordance through hard, sensor-triggered transitions. The skill layer of

Sec. 4–5 then rides on top, learning the per-segment precisions that reshape each phase and that select each phase's target by anticipating the next.

3.1 Task and platform

Experiments use a Kinova Gen3 7-DoF arm fitted with a Robotiq 2F-85 parallel gripper, simulated in Webots with full rigid-body physics and a wrist camera.

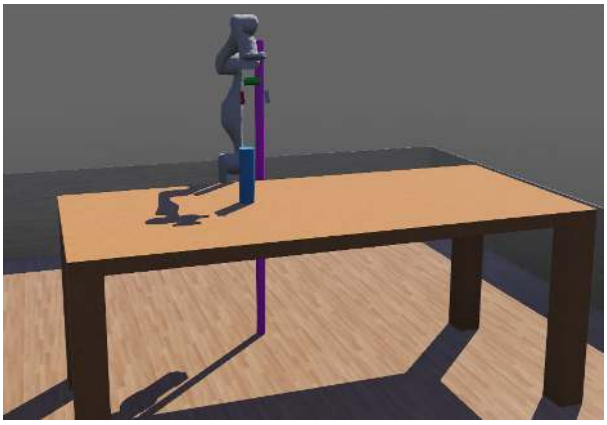


Fig. 1. Webots scenario for the experiments with the Kinova Gen3 arm.

The affordance is a *side-grasp pick-and-place* of an upright bottle standing on a table: from a rest pose the arm reaches a standoff beside the bottle, eases in and closes the fingers around the body, lifts it, carries it to a target spot, sets it down upright, releases, and withdraws to rest — one *episode*. The eight phases of this sequence (Sec. 3.4) group into the five graded motion segments — approach, insert, lift, place, retreat — over which skill is partitioned (Sec. 4), plus two ungraded commit events (closing the fingers, releasing) that carry no speed knob. Throughout the learning experiments the pick and place locations are fixed, so that any change across episodes is attributable to the agent's own refinement rather than to task variation; the object pose is read from the simulator as ground truth, and a fresh upright bottle is re-instantiated at the pick spot at the start of each episode.

3.2 Generative model

The agent's model factorises over a hidden joint configuration q (tool position $x = f(q)$ and orientation $R(q)$ by forward kinematics), an observation o, and a discrete policy π (the phase and the grasp azimuth):

$$p(o, x, \pi) = \underbrace{\sigma\big(-G(\pi)\big)}_{p(\pi)\ \text{policy prior}} \underbrace{\mathcal{N}\big(x; x^\star_\pi, \Pi_C^{-1}\big)\, e^{\,\alpha_o\, z_{\text{tool}} \cdot z^\star_\pi}}_{\tilde{p}(x|\pi)\ \text{preferred outcome}} \underbrace{\mathcal{N}\big(o; f(q), \Pi_s^{-1}\big)}_{p(o|x)\ \text{likelihood}}. \qquad (1)$$

A three-variable joint admits $3! = 6$ chain-rule factorisations, all algebraically equal. We write the one aligned with the *generative* (causal) direction $\pi \to x \to o$, whose standard form is $p(\pi)\, p(x \mid \pi)\, p(o \mid x, \pi)$: policy, then state, then observation. The third factor simplifies to $p(o \mid x)$ because an observation depends only on the current pose, $o \perp\!\!\!\perp \pi \mid x$ — a policy acts on the world *through* the state, never directly. Each factor is then a standard, physically meaningful mechanism — a controllable cause $p(\pi)$, a state preference $\tilde{p}(x \mid \pi)$, and a sensor model $p(o \mid x)$, recovering Eq. (1). The reverse orderings, such as conditioning the policy on the observation, would manufacture acausal factors with no generative

meaning and obscure the conditional independence we rely on. A reinforcement-learning controller rests on exactly that ordering: it learns a policy that maps observation to action, $o \to \pi$ — observation first, policy second — a discriminative control map rather than a generative model. Active inference instead keeps the causal direction $\pi \to x \to o$ and recovers the action by inference within it.

This form is therefore not arbitrary. The state prior is a *preference* conditioned on the policy, that is, a prior over the outcomes the agent expects to occupy, and this is what makes the model active rather than merely predictive. There is no transition term $p(x_{t+1} \mid x_t, a)$, because within-phase control is reactive and the temporal structure is carried entirely by the discrete π.

We now take the three factors in the order they appear in Eq. (1). The policy prior comes first. Here $\sigma(\cdot)$ is the softmax over the discrete policy set, so $p(\pi) = \sigma(-G(\pi)) = e^{-G(\pi)} / \sum_{\pi'} e^{-G(\pi')}$ places exponentially more mass on policies of lower *expected* free energy $G(\pi)$. This is the standard active-inference scoring of a policy by its anticipated cost, and in general its information gain. A policy here is a discrete choice: the phase and, at the grasp, the approach azimuth. We evaluate G one step ahead, and committing to the most probable policy, $\pi^\star = \arg\min_\pi G(\pi)$, is exactly the anticipatory selection of Sec. 5, with G approximated there by the manipulability surrogate of Eq. (12).

The preferred-outcome factor follows. It pairs a Gaussian position preference $\mathcal{N}(x; x_\pi^\star, \Pi_C^{-1})$ centred on the target $x_\pi^\star$, whose precision (inverse covariance) Π_C sets how tightly the agent prefers to sit on that target — its role in control is detailed in Sec. 3.3 — with an orientation term. That orientation term $e^{\alpha_o z_{\text{tool}} \cdot z^\star}$ is a von Mises–Fisher density, the unit sphere's analogue of a Gaussian. Its exponent is the cosine alignment of the achieved and desired approach axes, and its concentration α_o is an inverse angular variance; this is why the orientation cost takes the $1 - \cos$ form.

The likelihood closes the model. It is Gaussian, and with ground-truth perception ($\Pi_s \to \infty$) it does not vanish to a constant; it collapses to a Dirac delta $\delta\big(o - f(q)\big)$ that *pins* the state to the observation, removing perceptual inference and leaving only action. The noisy-perception case ($\Pi_s = 1/\sigma_{\text{obs}}^2$, a finite measurement precision) is treated in Sec. 7.

Free energy. Perception and within-phase action both minimise the *variational free energy* of this model,

$$F \;=\; \underbrace{D_{\text{KL}}\big[\, q(x) \,\big\|\, \tilde{p}(x \mid \pi) \,\big]}_{\text{complexity (preference divergence)}} \;\; \underbrace{-\, \mathbb{E}_q\big[\, \ln p(o \mid x) \big]}_{\text{accuracy}} \;\geq\; -\ln p(o \mid \pi), \tag{2}$$

a tractable upper bound on the surprise $-\ln p(o \mid \pi)$. Under exact perception the state belief collapses onto the observation, $q(x) = \delta\big(x - f(q)\big)$ (the Dirac limit above), the accuracy term is constant, and F reduces to the preference cost

$$F(q) \;=\; \tfrac{1}{2}\, \|x(q) - x_\pi^\star\|_{\Pi_C}^2 \;-\; \alpha_o\, z_{\text{tool}} \cdot z_\pi^\star \;+\; \text{const}, \tag{3}$$

whose negative gradient is exactly the preferred twist $\xi^\star = [\, v^\star; \omega^\star \,]$ realised by the controller of Sec. 3.3. Policy selection instead minimises the *expected* free energy $G(\pi)$ of the policy prior above, given concretely as the manipulability surrogate of Eq. (12) in Sec. 5.

3.3 Continuous controller

The within-phase controller turns a phase's precision-weighted pose preference into a joint-velocity command. The negative gradient of that preference is a desired spatial twist $\xi^\star = [\, v^\star;\, \omega^\star \,] \in \mathbb{R}^6$: a position term that pulls the tool toward the target $x^\star$ and decelerates to an exact, finite-time stop inside a deadband δ_x (the preference *tolerance width*; a tight δ_x encodes a high position precision Π_C), where a plain proportional law would instead settle only asymptotically; and an orientation term that rotates the tool frame to $R^\star$ along the shortest $SO(3)$ geodesic [8] by a proportional ramp of gain α, capped at $\omega_{\max}$ with deadband δ_θ. The gain and cap set the wrist *slew rate* and are skill-scaled in Sec. 5, so a confident agent reorients faster in free space.

Quadratic-program backend. The twist is resolved to joint velocities $\dot{q} \in \mathbb{R}^7$ by a strictly convex quadratic program (QP) over $(\dot{q}, \delta)$ with task slack $\delta \in \mathbb{R}^6$ [5],

$$\min_{\dot{q},\delta} \;\; \tfrac{1}{2}\big(\lambda^2 \|\dot{q}\|^2 + \delta^\top W_\delta\, \delta\big) + g^\top \dot{q} \quad \text{s.t.} \quad J\dot{q} - \delta = \xi^\star, \;\; l \le C\dot{q} \le u, \tag{4}$$

with $J = [J_v; J_\omega] \in \mathbb{R}^{6\times 7}$ the tool Jacobian, λ a damping that regularises near singularities, and $W_\delta = \mathrm{diag}(I_3, \rho I_3)$ down-weighting orientation slack ($\rho < 1$) so the solver never trades position to perfect an unreachable orientation. The linear term $g = -w(g_\mu \nabla\mu + g_e \nabla\phi_e)$ folds the redundancy objectives — Yoshikawa manipulability $\mu = \sqrt{\det JJ^\top}$ and an elbow-posture term ϕ_e — into the spare degree of freedom. With no active inequality and $\rho = 1$ the QP reduces to the standard damped-least-squares resolved-rate solve.

Constraints reshape each phase. The point of the QP form is that the same backend carries hard constraints *inline*, so each phase reshapes the controller to its own needs without changing the law. The inequalities $C\dot{q} \le u$ are collision and joint-limit velocity dampers [5] that ramp a robot point's approach speed toward an obstacle to zero at contact, a hard non-penetration guarantee ($\dot{q} = 0$ is always feasible). Phase-specific hard *equalities* reshape the motion further: the lift pins the angular velocity, $J_\omega \dot{q} = 0$, making the twist a *pure translation*, so the gripper rises carrying its exact orientation and cannot let the off-axis-held object pivot — the infinite-precision limit of the orientation preference, more robust than a soft weight that merely trades rise against level and stalls.

3.4 Discrete sequencer (formal structure)

The task is sequenced by a hybrid automaton $\mathcal{H} = (\mathcal{S},\, q,\, \{F_s\},\, \{\mathcal{G}_{s\to s'}\},\, \{R_{s\to s'}\})$ over phases $\mathcal{S} = \{$Tracking, Inserting, Closing, Lifting, PlaceMoving, PlaceLowering, PlaceReleasing, PlaceRetreating$\}$. Within phase s the flow F_s is the controller of Sec. 3.3 toward that phase's attractor $\xi_s^\star$ and gripper command; a transition fires when its guard $\mathcal{G}_{s\to s'}$ — a threshold predicate on the sensed state — becomes true. Its reset $R_{s\to s'}$ is the automaton's jump map: it leaves the continuous joint state untouched but instantaneously sets the auxiliary variables the next phase needs. On closing the gripper, for instance, it *latches the grasp frame* — the now-fixed gripper-to-object transform, so the bottle is thereafter carried rigidly — and on entering the place phase it *samples the place pose*

that becomes the next attractor $x^\star$. *Exactly one phase is active at any instant: the belief over phases is one-hot (all mass on a single phase) and the switches are hard.* What the agent adapts are the phase *knobs*, not the automaton's structure. We call the deterministic maps from the learned segment confidence c_k (Sec. 4) to those knobs the *knob dynamics*; they reshape the active phase as confidence grows,

$$\text{geometry (approach waypoint):} \quad x^\star_{\text{track}} = x_{\text{grasp}} + (x_{\text{standoff}} - x_{\text{grasp}})\,(1 - \kappa\, c_{\text{approach}}), \tag{5}$$

$$\text{cruise speed:} \quad v_{\max} = v_0\,(1 + g_v\, c_k), \tag{6}$$

$$\text{wrist slew (gain, cap):} \quad (\alpha, \omega_{\max}) = (\alpha_0, \omega_0)\,(1 + k_o\, c_{\text{approach}}), \tag{7}$$

$$\text{commit tolerance:} \quad \varepsilon_\theta = \varepsilon_{\theta 0}\,(1 + k_t\, c_{\text{approach}}). \tag{8}$$

As $c_{\text{approach}} \to 1$ the approach waypoint collapses from the standoff toward the grasp (5), while the cruise speed, the wrist slew rate, and the commit tolerance each rise linearly with confidence ((6)–(8)): a more confident agent travels faster, reorients sooner, and commits on a looser gate. The slew and tolerance knobs are deployed *anticipatorily* in Sec. 5. The guards $\mathcal{G}_{s \to s'}$ are fixed; only the timing, geometry, speed and tolerance move with c. This is therefore a finite-state machine with *learned knobs* and *hard switches*, not a continuous belief over phases; Sec. 8 discusses the migration.

4 Skill as Per-Segment Precision Accumulated by Doing

We take skill to be *confidence in the agent's own model* of the task — its precision, in the active-inference sense [3]. The model predicts the outcome of each attempt (the standoff is reached, the grasp holds, the object lands upright); how far the agent trusts those predictions is a model precision Π_m, normalised against the sensory precision Π_s of Sec. 3.2 into a confidence $c = \Pi_m/(\Pi_m + \Pi_s) \in [0, 1)$.

Crucially Π_m is *accumulated evidence*, not a hand-set schedule: each completed attempt confirms or contradicts the model's prediction and updates it as

$$\text{confirm:} \quad \Pi_m \leftarrow \Pi_m + u\,\eta, \qquad \eta \in (0, 1] \tag{9}$$

$$\text{surprise:} \quad \Pi_m \leftarrow \gamma\,\Pi_m, \qquad \gamma \in (0, 1), \tag{10}$$

where $u > 0$ is the base evidence a clean outcome is worth and η scales it by a *quality* measuring how closely the outcome matched the prediction (a small terminal error and a full lift give $\eta \approx 1$, a partial outcome less). Over n cleanly confirmed attempts c follows the saturating curve $c \approx n/(n + \Pi_s)$, rising fastest when the model is least trusted and levelling as evidence accrues. The learned c then *scales the action* through the schedules of Sec. 3.4 — the more the model has been confirmed, the more the agent acts on it — with no reward and no offline policy search. The process is *self-calibrating*: a surprise deflates Π_m, so c rises only as fast as outcomes keep confirming and over-trust that causes a failure is pulled back; and because Π_m is persisted, skill is durable across sessions.

Per-segment partition. A single global confidence buys only one operating point on the speed–reliability trade-off, yet the segments of a pick-and-place have opposite needs: free-space transport can cruise while the contact-rich segments (insertion, set-down) must stay careful. We therefore *partition* the precision, giving each motion segment $k \in \{\text{approach, insert, lift, place, retreat}\}$ its own $c_k = \Pi_m^{(k)}/(\Pi_m^{(k)}+\Pi_s)$, credited from its *own* outcome: the approach from its terminal standoff error, the insert from reaching contact, the lift from the achieved rise fraction, place and retreat from the object's post-motion tilt (upright vs. tipped). The active segment selects which c_k drives the current knob, so every schedule becomes segment-local. This yields three properties: segments learn *in parallel* (one episode feeds every segment it traverses, so none is starved); credit *and* caution are *localised* (a surprise deflates only the failing segment, so one miss no longer discards the skill earned on the others); and skill *self-allocates* where it is earned. The discrete commit events (closing the gripper, releasing) carry no speed knob and remain hard.

5 Anticipation: Each Phase Frames the Next

A segment is *useless* if it leaves the arm where the next cannot proceed. The policy principle minimises *expected* free energy over the sequence, choosing each phase's preference so the next has low surprise. Two instances reuse the accumulated precision.

The grasp anticipates the lift. A side grasp is near-singular for the straight-up lift over roughly half the approach directions, so μ collapses on the rise and the rigidly held bottle twists $\sim$40° until it tips, though the grasp looked clean. The bottle being a symmetric cylinder, the approach *azimuth* is free, so we score candidates with a fast inverse kinematics (IK) and pick the one whose manipulability survives *both* grasp and lift,

$$\theta^\star = \arg\max_\theta \ \min\big(\mu(q_{\text{grasp}}(\theta)),\, \mu(q_{\text{lift}}(\theta))\big). \tag{11}$$

Equation (11) is a tractable surrogate for a one-step expected-free-energy choice over $\{\theta\}$: under a quadratic twist preference the expected surprise of not realising the commanded motion is monotonically proxied by $1/\mu$, so the worst μ along the grasp→lift policy gives

$$G(\theta) \ \approx \ -\beta\, \min\big(\mu(q_{\text{grasp}}(\theta)),\, \mu(q_{\text{lift}}(\theta))\big) + \text{const}, \tag{12}$$

minimised exactly by (11), so the grasp trades a hair of grasp-optimality for a *feasible* lift. Because the IK ranking can be over-optimistic, a standoff guard recomputes the *realised* $\mu = \sqrt{\det JJ^\top}$ from the measured joints (IK-free) and rotates to the next azimuth if it has fallen too low. With the pure-translation lift constraint of Sec. 3.3, this turns a near-always-failing lift reliable (Sec. 6).

The approach hands off to the insertion. The approach reaches the standoff fast, but the wrist then crawls the last degrees of orientation to a tight gate — the episode's largest cost. Reorientation is contact-free, so the wrist-slew schedule

(7) *front-loads* it to finish during travel, and the commit-tolerance schedule (8) lets the approach commit looser, handing its residual cant to the insertion to finish *while travelling*. Both are safe because the azimuth choice made the lift insensitive to that cant.

6 Experiments

We evaluate on a Kinova Gen3 7-DoF arm with a Robotiq 2F-85 gripper in Webots, performing a side-grasp pick-and-place of an upright bottle with fixed pick and place locations, so that variation across episodes reflects learning rather than task geometry. The within-phase controller is the NEO-style QP backend of Sec. 3.3. A *round* is ten episodes; the precision parameters are $\Pi_s = 2$, $u = 1$, $\gamma = 0.5$, and Π_m is persisted to disk between sessions, so a re-launched round resumes already skilled ($c \approx 0.61$ at the first commit) rather than at $c = 0$ — the learned quantity is durable, not a within-session artefact.

6.1 Per-segment divergence and decoupling

The five segment confidences are learned in parallel and diverge by their own outcome (Fig. 2): `insert` and `retreat` confirm cleanly every episode (binary success) and saturate fastest, while `lift` — which on the full system succeeds every time but is graded by the partial-rise fraction, a continuous quality below one — accrues precision the slowest and stays lowest. The ordering `insert` $\approx$ `retreat` $>$ `place` $>$ `approach` $>$ `lift` is reproducible across all three cold-start rounds (the min–max band in Fig. 2 is negligible) and the final precisions are nearly identical round-to-round (Π_m for `lift`$\approx$5.2, `approach`$\approx$6.3, `place`$\approx$7.3, with `insert`/`retreat` saturating at $\approx$10). The partition's defining property is that caution is localised: because the deflation rule (10) acts only on the failing segment, where a single global confidence would lose a slice of *all* its skill to one miss (we observed a $0.58 \to 0.52$ dip), the partition deflates only the culprit and leaves the others' earned precision intact — marking `lift`, the most conservatively graded, as the delicate segment the anticipatory grasp of Sec. 6.2 then targets.

6.2 Anticipation makes the lift reliable

That `lift` is the most delicate segment has, with a *greedy* grasp, a purely geometric cause rather than a learning-rate one: the lift drove into a wrist near-singularity, μ collapsing from ≈ 0.09 at the grasp to < 0.005 during the rise, so the rigidly held bottle was twisted to 40–55° and tipped — *bimodally* and independently of confidence (clean on some episodes, failing on most), which is why the greedy system's `lift` took the misses. The lift-aware azimuth of Eq. (11) removes the cause: the chosen approach keeps $\mu \approx 0.09$ across both the grasp and the lift, and the pure-translation constraint then keeps the gripper rigid. The lift then goes from near-always failing to clean on every trial (8/8 in a first verification, and 30/30 across the three cold-start rounds, the object held

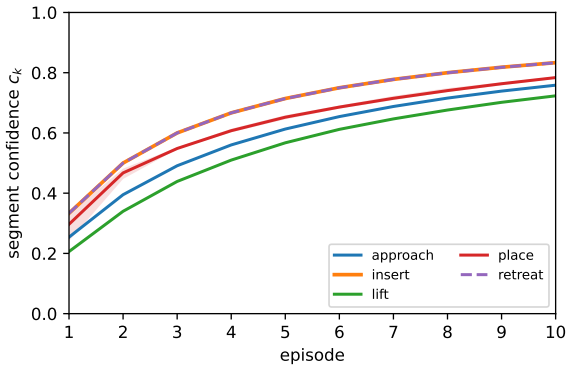


Fig. 2. Per-segment confidence c_k vs. episode (mean of three cold-start rounds; band = min–max, negligible). Segments diverge by their own outcome: `insert`/`retreat` saturate (*retreat* dashed drawn over the coincident *insert*), `lift` stays lowest (graded by partial-rise fraction).

within 1° of upright), and the standoff μ-guard absorbs the rare azimuth on which the inverse-kinematics ranking is over-optimistic. Anticipation thus *cures* the geometric weakness that the partition could only *flag*.

6.3 Whole-episode execution time vs. episode

The original limitation of the partition-only system was that only the c-controlled portion of the motion sped up, while the *whole*-episode time — dominated by the orientation-bound approach — barely moved. With the anticipatory approach-to-insertion hand-off of Sec. 5, the approach is no longer the floor, and the whole episode falls with skill: from 10.1 s at the cold first episode to 5.3 s by the tenth, a ∼48% reduction (mean of three rounds), with every episode successful (Fig. 3). The duration curve is the near-perfect mirror of the rising mean confidence $\bar{c}$ plotted alongside it, and starts at the novice value — the slew and tolerance schedules are inert at $c = 0$ — so the gain is unambiguously the learned skill being deployed anticipatorily, not a re-tuned constant. The pick portion (rest→grasp) carries most of the reduction, consistent with the approach being the segment the anticipatory schedules touch.

6.4 Ablation: what each component contributes

To attribute the gains we toggle each component independently at the fixed pick spot, ten cold-start episodes each (Table 1); all variants succeed 10/10 here, so this experiment isolates *speed*, with reliability held uniform. Three results stand out, and the first qualifies our own framing. (i) Anticipation removes a singularity-induced slowdown — it is not itself a speed mechanism. The greedy grasp at this pose is not infeasible (it succeeds 10/10), but it sits at low manipulability ($\mu \approx 0.05$ against 0.09 for the lift-aware azimuth), and a *damped* resolved-rate controller traverses a near-singular configuration slowly: the achiev-

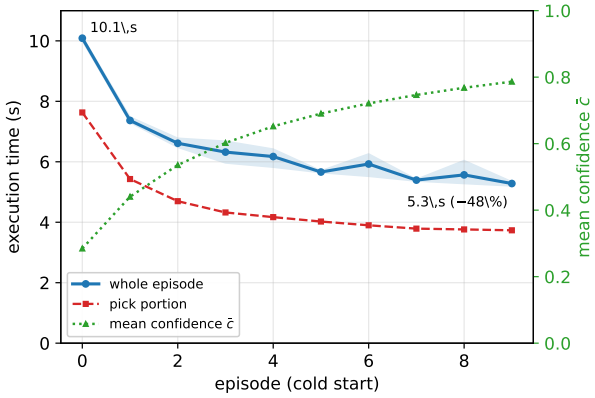


Fig. 3. Whole-episode and pick-portion time vs. episode (mean of three rounds; band = min–max), with mean confidence $\bar{c}$ (right axis). The full system shortens the episode ∼48% as $\bar{c}$ rises, success preserved; the curves mirror each other.

able Cartesian speed is throttled as μ falls. Anticipation's 24.6 → 10.0 s is therefore the removal of that penalty, not an acceleration — the *same* manipulability that, away from this pose, separates a feasible lift from a failing one here separates a fast traverse from a crawling one. Consistently, the anticipation-only time (10.0 s, constant since its schedules are inert at $c = 0$) coincides with the *cold first episode* of the full system (Fig. 3); the precision layer then supplies the drop along the round — from that 10.1 s cold episode to 5.3 s by the tenth (Sec. 6.3), i.e. the 6.9 s per-round mean in Table 1. So the *genuine, learned* speed-up — commanding faster motion as confidence grows — is precision's: *anticipation clears the way, precision accelerates along it.* (ii) The per-segment partition beats a single global confidence (18.6 vs. 21.0 s): localised caution does not throttle the whole motion for one segment's risk. (iii) At this easy, always-succeeding task, precision is indistinguishable from a success-rate–scheduled gain (6.4 vs. 6.9 s for the full system): a Beta success-rate driving the *same* knobs is, if anything, marginally faster because it starts optimistic rather than cautious. The purpose of this baseline is not to claim superiority on an always-successful task, but to show that precision does not derive its benefit from faster gain scheduling: its distinguishing behaviour emerges only when failures occur and confidence must be localised and retracted. Precision's distinctive value — earning caution, localising it, and retracting it on surprise — is therefore not observable where nothing fails; exhibiting it requires a failure-prone regime (perception noise, perturbed poses), which we leave to future work. We state this candidly: *anticipation* secures feasibility — and, where the greedy pose is near-singular, removes the resulting slowdown — while the *precision* layer supplies the learned, reward-free speed-up and localises caution; at an easy task that speed-up is matched by a hand-tuned success schedule, its distinctive caution-earning role awaiting a failure-prone test.

Table 1. Component ablation at the fixed pick spot (ten cold-start episodes each; all 10/10 successful, so reliability is uniform and the comparison is on whole-episode time). Each entry is the mean whole-episode time over those ten episodes; for the full system this 6.9 s mean corresponds to the per-episode learning curve falling $10.1 \rightarrow 5.3$ s (Sec. 6.3). Anticipation is the dominant speed contributor; per-segment precision beats global; and at this easy task precision matches a success-rate–scheduled gain on the same knobs. "Δ" is the reduction vs. the fixed-schedule base.

variant	episode time (s)	Δ vs. base
Base FSM (fixed schedule)	24.6	—
+ precision (global)	21.0	−15%
+ precision (per-segment)	18.6	−24%
+ anticipation (no precision)	10.0	−59%
Full (per-segment + anticipation)	6.9	−72%
success-rate gain (+ anticipation)	6.4	−74%

7 Discussion and Limitations

The system remains, structurally, a finite-state machine with hard, threshold-triggered switches: the partition upgrades the *learning substrate* and anticipation upgrades *what each phase aims at*, but neither yet makes the inter-phase *transitions* continuous: the set-down segments still stop and restart between vias, which is the residual whole-episode time that the continuous field proposed below targets. Retaining hard switches buys interpretability and a clean safety story at that cost. The anticipatory choice is a one-step arg max, sufficient at a boundary but not global planning; and the learning is *exploit-only* — each c_k converges to the sustainable point under the *current* schedules, never probing a still-faster one. Finally, object pose is read as simulator ground truth at the control rate, so perception cost is factored out of our timings — on hardware a look-up (tens of ms for segmentation and pose estimation) would itself bound the loop, capping it well below the controller's rate. The same precision principle dissolves this: as Π_m grows a fresh observation carries vanishing epistemic value, so the agent can act on the *model* between sparse look-ups (open-loop with surprise-triggered revert), decoupling the perception rate from the control rate. That, with sim-to-real transfer and the precision-weighted observation fusion it relies on, is left to future work.

8 Conclusion and Future Work

A hand-specialised manipulation skill can improve itself — faster, at least as reliable — purely by being executed, with no reward and no offline policy search, by operationalizing skill as accumulated model precision: partitioned per segment it localises caution, and deployed anticipatorily it clears the near-singular floor that capped the approach, after which the learned precision lifts the whole-episode time ∼48% along the cleared path. Anticipation clears the way; precision accelerates along it.

The natural next step is structural: the per-segment precisions are exactly what a *continuous* sequencer would consume. Replacing the hard-switch automaton with a via-point preference field [6,9], in which each via's precision sets a terminal speed between stopping and passing through ($\lambda = e^{-\Pi/\Pi_{\text{ref}}}$), would turn the remaining discrete commits into soft transitions and let coarticulation emerge across the whole sequence. A failure-prone regime — perception noise, perturbed poses — is also where precision's localised, retractable caution should finally separate it from a plain success-rate schedule.

Acknowledgments. This work was co-funded by the FEDER Project 0124_EUROAGE_MAS_4_E under the POCTEP Programme 2021–2027, and by the R&D&i project PID2022-137344OB-C31, supported by MICIU/AEI/10.13039/501100011033 and "FEDER, Una manera de hacer Europa". Additional co-funding was provided by the European Union through the European Regional Development Fund (85%) and by the Junta de Extremadura (Grant GR24194). The managing authority is the Ministerio de Hacienda (Spain). This work was also supported by the SWEET Project, a Horizon Europe Marie Skłodowska-Curie action (Agreement No. 101168792).

References

1. Baioumy, M., Duckworth, P., Lacerda, B., Hawes, N.: Adaptive manipulator control using active inference with precision learning. In: UKRAS Conference on 'Robots into the real world' (2020), also available as a workshop/technical report
2. Bristow, D.A., Tharayil, M., Alleyne, A.G.: A survey of iterative learning control. IEEE Control Systems Magazine **26**(3), 96–114 (2006)
3. Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., Pezzulo, G.: Active inference: A process theory. Neural Computation **29**(1), 1–49 (2017). https://doi.org/10.1162/NECO_a_00912
4. Friston, K., Rigoli, F., Ognibene, D., Mathys, C., Fitzgerald, T., Pezzulo, G.: Active inference and epistemic value. Cognitive Neuroscience **6**(4), 187–214 (2015). https://doi.org/10.1080/17588928.2015.1020053
5. Haviland, J., Corke, P.: NEO: A novel expeditious optimisation algorithm for reactive motion control of manipulators. IEEE Robotics and Automation Letters **6**(2), 1043–1050 (2021). https://doi.org/10.1109/LRA.2021.3056060
6. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: Learning attractor models for motor behaviors. Neural Computation **25**(2), 328–373 (2013). https://doi.org/10.1162/NECO_a_00393
7. Lanillos, P., Meo, C., Pezzato, C., Meera, A.A., Baioumy, M., Ohata, W., Tschantz, A., Millidge, B., Wisse, M., Buckley, C.L., Tani, J.: Active inference in robotics and artificial agents: Survey and challenges. arXiv preprint arXiv:2112.01871 (2021)
8. Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press (1994)
9. Paraschos, A., Daniel, C., Peters, J., Neumann, G.: Probabilistic movement primitives. In: Advances in Neural Information Processing Systems (NeurIPS) (2013)
10. Pezzato, C., Çatal, O., Van de Maele, T., Pitliya, R.J., Verbelen, T.: Mobile manipulation with active inference for long-horizon rearrangement tasks. arXiv preprint arXiv:2507.17338 (2025)
11. Pezzato, C., Ferrari, R.M.G., Hernández Corbato, C.: A novel adaptive controller for robot manipulators based on active inference. IEEE Robotics and Automation Letters **5**(2), 2973–2980 (2020). https://doi.org/10.1109/LRA.2020.2974451

Advancing the Assistive Social Robotics Fleet of the EuroAGE+ Network: A Cross-Platform Analysis

Pedro Núñez[1], Rui P. Rocha[2], and Paulo J. S. Gonçalves[3]

[1] RoboLab, University of Extremadura, Cáceres, Spain
[2] Institute of Systems and Robotics, University of Coimbra, Coimbra, Portugal
[3] Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal

Abstract. Population ageing in the Iberian cross-border EuroACE region motivates the deployment of socially assistive robots (SARs) that support physical, cognitive, and socio-emotional dimensions of active and healthy ageing. Within the Interreg Spain–Portugal project EuroAGE+, three partner institutions consolidate a heterogeneous robot fleet built and validated across a decade of preceding projects. This paper presents an analytical, cross-platform synthesis of the recent advances introduced by these platforms—GrowMu (ISR/UC), the EBO/EBOv2/Shadow family (UEX), and Amiga (IPCB)—during EuroAGE+. We characterise hardware and software trajectories and identify convergent design trends: a systematic migration to ROS 2, the adoption of cloud- and LLM-based conversational stacks, an emphasis on low-cost additively manufactured embodiments, the integration of adaptive cognitive architectures, and a growing concern with personalised and ethically grounded interaction. We argue that these independent platforms are converging toward a common methodological substrate, and we discuss the implications for the project's stated goal of deploying a second, ethics-aware fleet of assistive robots. Two recently validated interaction systems—LLM-driven personalised storytelling on GrowMu and a CORTEX-based, LLM-powered cognitive-therapy system on EBO—ground the analysis in pilot evidence gathered with older adults and occupational therapists.

Keywords: Socially assistive robotics · Active ageing · Human–robot interaction · Cognitive stimulation · ROS 2.

1 Introduction

The demographic shift toward an older population imposes sustained pressure on care systems and has accelerated research into technologies that prolong autonomous, healthy life. Socially assistive robots (SARs)—robots that "interact with humans and with each other in a socially acceptable manner, conveying intentions in a human-perceptible way, and achieving goals in collaboration with other agents" [1]—are a central instrument of this effort, acting as the physical and interactive substrate of broader digital assistance ecosystems.

The Interreg Spain–Portugal (POCTEP) project EuroAGE+ continues a line of cross-border collaboration in the EuroACE region that began with EuroAGE (2017–2020) and EuroAGE2 (2022–2023). Its second activity is explicitly devoted to consolidating an assistive social-robotics network and to deploying a *second fleet* of social robots endowed with personalised and ethically grounded assistance capabilities. The activity targets three coupled objectives: improving robot and environment perception, advancing personalised social behaviour and semantic knowledge representation, and aligning human–robot interaction with contemporary ethical standards for autonomous robotic systems.

This paper contributes an analytical, cross-platform account of how three independently developed robot platforms have advanced within EuroAGE+. Rather than describing each system in isolation, we treat the fleet as a single evolving artefact and ask what design trends are shared across partners with distinct engineering traditions. Section 2 frames the fleet. Sections 3–5 analyse the three platforms. Section 6 examines, in depth, two interaction systems that have been validated with end users and provides quantitative evidence. Section 7 synthesises convergent trends, and Section 8 concludes.

2 The EuroAGE+ Assistive Robot Fleet

The fleet spans three institutions and four platform lineages. The Institute of Systems and Robotics of the University of Coimbra (ISR/UC) contributes *GrowMu*, a mature platform in service since 2011. The University of Extremadura (UEX), through its RoboLab group, contributes the compact *EBO* and *EBOv2* therapy robots and the mobile human-following platform *Shadow*. The Polytechnic Institute of Castelo Branco (IPCB) contributes *Amiga*, the successor to its earlier *Rato* and *Amigo* prototypes. The platforms differ markedly in size, mobility, cost, and computational provisioning, yet they share a common application target: physical, cognitive, and socio-emotional stimulation of older adults in day centres, residential care, and, increasingly, at home.

3 GrowMu (ISR/UC): From Bespoke Pipelines to Cloud-Native Dialogue

3.1 Heritage and Hardware

GrowMu originated in the European project SocialRobot (2011–2015), which delivered the platform's ROS 1 drivers and an early empathic interaction system based on facial-expression and gesture analysis [4]. Subsequent projects extended its repertoire: GrowMeUp added natural-language voice interaction, autonomous navigation, scheduled behaviour, fall detection, medication reminders, and object localisation; the first EuroAGE project embedded GrowMu in a distributed multi-agent system—virtual agent, smart-camera network, and robot—for physical-activity promotion grounded in technological persuasion [3]; and ActiVas introduced emotion recognition with behavioural adaptation and movement-

based serious games, including a game for sarcopenia assessment [5]. The platform is a differential-drive unit equipped with a 12-sonar ring, ground-facing infrared sensors, a Hokuyo laser range finder, RGB and RGB-D cameras, a touchscreen, expressive LED arrays, and an Intel i7 Mini-ITX controller.

3.2 Advances During EuroAGE+

Within EuroAGE+, ISR/UC reframed GrowMu's interaction stack around modern language technologies [8]. Speech-to-text and text-to-speech functions were delegated to cloud service providers, yielding a measurable improvement in speech naturalness and response latency relative to the GrowMeUp-era pipeline [6]. Building on this stack, the team developed an LLM-based interactive storytelling system in which a therapist parameterises a narrative that the robot then co-constructs with the user through natural language, synchronising synthesised speech with LED lip animation [7]. A key engineering milestone was the migration of the system from ROS 1—whose official support ended in May 2025—to ROS 2 (Jazzy distribution), ensuring future maintainability; the source code was released openly [7].

The storytelling system (Fig. 1) is organised into a dialogue interface, a dialogue-management module (DMM), and a therapist interface. The DMM queries an OpenAI assistant (model `gpt-3.5-turbo-1106`) instructed to produce simple interactive stories of at most 80 words per turn, two choices per continuation, and a six-interaction ceiling, with vocabulary matched to the user's cognitive level [7]. The dialogue interface is provider-agnostic: TTS can be sourced from Microsoft Azure, Google Cloud, or ElevenLabs, while STT currently uses Azure, but may be easily extended to support other cloud service providers. Lip synchronisation is realised through two strategies—direct mapping of Azure's IPA-based viseme stream to the robot's mouth-LED expressions, and, for providers that do not provide directly visemes, a lightweight audio-energy analysis that maps signal energy to discrete degrees of mouth opening for real-time synchronisation [7]. A TCP/IP therapist GUI lets occupational therapists manage user profiles, set a story theme, supervise sessions, and receive transcripts, with a server preserving user records across institutions.

4 EBO, EBOv2, and Shadow (UEX): Low-Cost Embodiments and Adaptive Cognition

4.1 Heritage and Hardware

RoboLab has built more than ten robots since 1995, and EBO descends from this line through the regional project ExtendAGE and the EuroAGE/EuroAGE2 initiatives, where it provided cognitive and emotional stimulation. Two further projects shaped the current generation: CAMPERO introduced a self-adaptive cognitive architecture—reused in EuroAGE+—integrating users, intelligent environment, and assistive robots; and a proof-of-concept project produced a Shadow variant capable of social following and navigation in care settings.

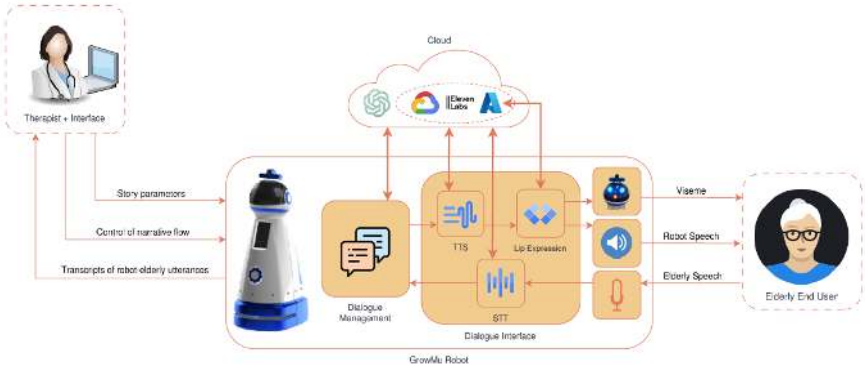


Fig. 1. Architecture of the GrowMu interactive storytelling system ([7]).

EBO is a compact, ergonomic therapy robot built around a Raspberry Pi 3B+, with an RGB camera on a servo mount, five VL53L0X laser sensors, a resistive PiTFT display, and differential DC drive. EBOv2 upgrades the computer to a Jetson Xavier NX, adds LiDAR, a wide-angle camera, a 7-inch touchscreen, and LED-based emotional expression, and is printed in recycled PLA for economical, scalable production. Shadow is an advanced mobile platform with omnidirectional Mecanum kinematics, dual RoboSense 3D LiDARs, a 360° Ricoh Theta Z1 camera, and an NVIDIA Jetson Orin running deep networks for human detection and tracking, supported by a 48 V battery providing at least seven hours of autonomy.

4.2 Adaptive Software and Validation

EBO runs a self-adaptive cognitive system built on the CORTEX architecture [14], whose shared working memory—a directed graph whose nodes denote ontology concepts and edges their relations—lets independent software agents read and update the robot's world model and trigger one another [13]. The interaction stack is realised as cooperating agents: an automatic-speech-recognition agent based on Whisper [15]; a text-to-speech agent based on MeloTTS [16], which synthesises expressive multilingual speech in real time on CPU; a dialogue-manager agent driven by a large language model (Llama [17] or GPT-class); a prompt-generator agent that fuses user, environment, and game state into LLM prompts; and a player/game-data agent embodying the therapist-in-the-loop. The therapist configures the session through a GUI; the system then runs autonomously and returns a session summary (Fig. 2).

The current cognitive-therapy use case implements three games for verbal training in Activities of Daily Living (ADL): *ADL Sequences* (ordering the steps of a task), *True or False of ADLs*, and *The Shopping* (identifying an out-of-place item and computing a total cost), exercising sequencing, comprehension, planning, and numeracy [13]. A separate EuroAGE+ pilot extended this line

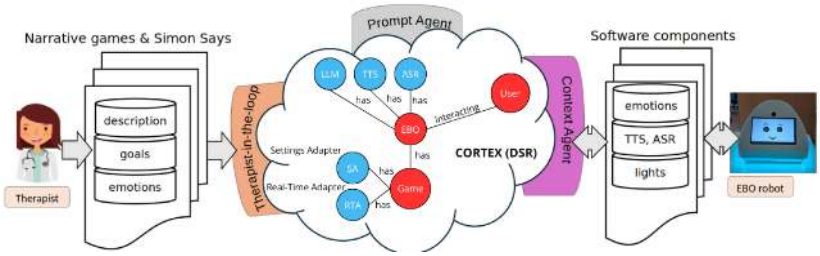


Fig. 2. Agent organisation of the EBO cognitive-therapy system on the CORTEX working memory ([13]).

Table 1. LLM benchmark for the EBO ADL-sequence game (after [13]). GPT-3.5 was selected for user-acceptance testing.

Model	Resp. time (s)	Cognitive adaptation	Personalisation	Discourse continuity
Llama 3 (8B, A100)	14.3	Full	High (name, gender)	None (no memory)
ChatQA-1.5-8B	33.2	None	Minimal	Short-term; fails after 3–4 turns
GPT-3.5	5.1	Full	High (name, gender)	Complete (recalls past turns)

with four adaptive serious games (Storytelling, Pasapalabra, Simon Says, and Adaptive Conversation): a first session at the Aztide day centre (Cáceres) exercised Storytelling with EBOv1, and a six-week study across three residential centres enrolled 35 older adults with mild-to-moderate cognitive impairment, alternating expressive and neutral styles under an ethics-approved protocol; users preferred expressive styles and EBOv2 and showed gradual cognitive gains in memory- and decision-intensive games.

A controlled benchmark of three LLMs for the ADL-sequence game (Table 1) informed the deployment choice: GPT-3.5 offered the lowest latency and the only complete discourse continuity, and was selected for user-acceptance testing, with Llama 3 retained as a promising on-premises alternative [13].

5 Amiga (IPCB): A ROS 2 Assistant with Modular Perception

5.1 Heritage and Hardware

IPCB's trajectory began with *Rato*, a tele-operable, Raspberry-Pi/Arduino arena robot for cognitive games validated in two Misericórdia institutions [9], and

Amigo, a mobile manipulator endowed in EuroAGE2 with smart-home task planning [10], ontology-based reasoning for elderly care [11], and a behaviour-tree control architecture for robust failure recovery [12]. The current platform, *Amiga*, integrates an RPLIDAR A2M12 and an Intel RealSense D435 for 2D/3D mapping, a LifeCam HD3000 for video, an omnidirectional microphone and JBL speaker, a 7-inch capacitive touchscreen serving as both interface and animated face, four Mecanum wheels with a 9-DoF IMU, and a dual-compute layout pairing a Raspberry Pi 4 with an Intel Core i7 mini-PC running ROS 2, backed by power-bank energy provisioning for up to eight hours of operation.

5.2 Software Modules

Amiga's capabilities are organised as discrete ROS 2 nodes: an Arduino-based environmental-sensing module (temperature, humidity, gas leaks); MediaPipe-based face detection and a dedicated recognition node for personalised interaction; a geometric facial-interface node with audio-synchronised mouth motion; an eye-tracking node that orients the gaze toward the person of interest; Google Cloud STT and TTS nodes with noise-adaptive capture; a reality-orientation questionnaire module for periodic well-being assessment; and a responsive web portal that lets caregivers locate the resident, launch video calls, and assume manual control in emergencies.

6 Evaluated Human–Robot Interaction Systems

Two EuroAGE+ interaction systems have advanced beyond prototype description to validation with users, and together they illustrate the convergence argued in this paper from complementary angles: an open-ended *storytelling* system on GrowMu [7] and a structured *cognitive-therapy* system on EBO [13]. Both pair a cloud or local LLM with neural ASR/TTS, keep a therapist in the loop, and personalise content to the user's cognitive level, yet they differ in robot embodiment, software substrate, and interaction format. Table 2 contrasts them.

6.1 Storytelling on GrowMu: Usability and Acceptance

The storytelling system was pilot-tested at a day centre run by Cáritas Diocesana de Coimbra (Central Portugal). Occupational therapists administered the Mini-Mental State Examination [18] and admitted only candidates scoring ≥ 15, yielding 15 older participants; two therapists conducted the sessions (Fig. 3). Each session lasted 10–15 minutes, of which the interactive narrative occupied roughly 7 minutes, followed by a questionnaire adapted from the System Usability Scale [19] into matched 10-item, 5-point instruments for elderly users and therapists. Among elderly users, items related to willingness to use, ease of understanding, sustained interest, well-being, and comfort all averaged above 4, with interaction quality rated the highest; comprehension of the robot's speech scored lower (3.73 ± 1.33), attributed by the authors to hearing acuity, and story-following

Table 2. Comparison of the two validated EuroAGE+ HRI systems.

Dimension	**GrowMu storytelling (ISR/UC)** [7]	**EBO cognitive therapy (UEX)** [13]
Robot	GrowMu (mobile column SAR)	EBO (compact desktop, <15 cm, <1 kg)
Software base	ROS 2 (Jazzy); open source	CORTEX architecture (shared working memory)
LLM	OpenAI `gpt-3.5-turbo-1106`	Llama 3 / GPT-3.5 (benchmarked)
ASR (STT)	Microsoft Azure	Whisper (OpenAI)
TTS	Azure / Google / ElevenLabs (pluggable)	MeloTTS
Expressivity	Mouth-LED viseme lip-sync	Screen/LED facial expressions
Interaction	Interactive personalised storytelling	Cognitive games (ADL sequences, true/false, shopping)
Personalisation	User profile + MMSE-based cognitive level	User profile + cognitive level
Therapist role	In-the-loop GUI: theme, supervision, transcripts	In-the-loop GUI: game config, session summary
Autonomy	Autonomous delivery, therapist supervises	Autonomous session, therapist summary

showed the most variance (3.87 ± 1.25), linked to theme-dependent narrative coherence. Therapists rated the configuration interface, usability without technical support, cognitive-level adaptation, and therapeutic usefulness highly. The authors report support for both hypotheses—high perceived usability (H1) and high therapist acceptance (H2)—while noting the small therapist sample [7].

6.2 Cognitive Therapy on EBO: User Satisfaction

The EBO system was assessed with a Likert-scale questionnaire after participants played a selection of the ADL games using the full agent stack [13]. Clarity of instructions and ease of understanding the robot's questions both averaged 4.7 ± 0.48, and willingness to recommend the activities was highest at 4.8 ± 0.42, indicating strong satisfaction. The weakest item was the perception that EBO understood spoken answers (3.9 ± 1.1), pointing to speech-recognition variability, particularly for some voices, as the principal target for improvement. Table 3 summarises headline results from both studies.

Taken together, the two studies converge on the same picture: LLM-driven, therapist-supervised interaction is well accepted and perceived as usable by both older adults and caregivers, while *speech recognition* remains the common

Table 3. Headline evaluation results of the two validated HRI systems.

System	Participants	Instrument	Selected results
GrowMu story-telling [7]	15 elderly (MMSE $\geq$ 15) + 2 therapists	Adapted SUS, 5-pt Likert	Usability items $>$ 4; comprehension 3.73 ± 1.33; following 3.87 ± 1.25; H1 & H2 supported
EBO cognitive therapy [13]	10 volunteers	9-item, 5-pt Likert	Clarity 4.7±0.48; recommend 4.8 ± 0.42; speech recognition 3.9 ± 1.1

Fig. 3. Validation sessions of the two interaction systems with older adults [7] [13].

bottleneck—comprehension of the robot's speech on GrowMu and recognition of the user's speech on EBO are, in each case, the lowest-scoring dimension.

7 Cross-Platform Discussion

Although the three partners pursued independent engineering paths, their EuroAGE+ contributions exhibit striking convergence (Table 4). Five trends stand out.

Migration to ROS 2. The end of ROS 1 support drove GrowMu and Amiga to a ROS 2 substrate, while UEX layered a self-adaptive cognitive architecture atop a comparable component model. The fleet thus shares a maintainable, node-oriented software foundation.

Cloud- and LLM-based dialogue. All three platforms replaced earlier rule- or command-based interaction with conversational stacks combining ASR, large language models, and neural TTS. This reflects a field-wide shift from scripted dialogue toward open-ended, personalised conversation as the principal interaction modality. The choice of model is, however, consequential: the

Table 4. Comparative summary of the EuroAGE+ platforms.

Aspect	**GrowMu (ISR/UC)**	**EBO family (UEX)**	**Amiga (IPCB)**
Embodiment	Differential-drive column	Compact desktop (EBO); omnidirectional mobile (Shadow)	Mecanum mobile base
Computer	Intel i7 Mini-ITX	Jetson Xavier NX / Orin	i7 mini-PC + RPi 4
Perception	Sonar, IR, LiDAR, RGB-D	VL53L0X / LiDAR, 3D LiDAR, 360° camera	RPLIDAR + RealSense
Middleware	ROS 1 → ROS 2	CORTEX / MAPE-K	ROS 2
Dialogue	Cloud STT/TTS, LLM storytelling	Whisper, GPT-class LLM, MeloTTS	Google STT/TTS
Production	Bespoke	3D-printed, low-cost	Modular, off-the-shelf
Validation	Real care settings	2 pilots, 35 users	Misericórdia sessions

EBO benchmark (Table 1) shows order-of-magnitude latency differences and, critically, that discourse continuity—recalling earlier turns—is not guaranteed across models, which directly shapes the coherence of multi-turn therapy.

Low-cost, additively manufactured embodiments. UEX's recycled-PLA EBOv2 and Shadow, and IPCB's modular Amiga, prioritise economical, scalable fabrication—directly serving the project goal of deploying low-cost prototypes that meet the minimum requirements for residential deployment in the EuroACE region.

Emotional expressivity and personalisation. From GrowMu's LED faces to EBOv2's expressive animations and Amiga's reactive facial interface, expressivity is treated as a first-class requirement, corroborated by the UEX pilot's finding that users prefer expressive interaction styles.

Ethics and real-world validation. The platforms increasingly embed ethical and semantic reasoning—ontology-based elderly-care reasoning on Amiga and the self-adaptive architecture on EBOv2—consistent with the project's adoption of international standards for ethically aligned autonomous robotic systems, and all report validation with end users rather than laboratory-only evaluation. The two pilots analysed in Section 6 make this concrete, reporting high usability and caregiver acceptance (Table 3); they also expose a shared limitation—speech recognition is the lowest-rated dimension in both—which we identify as the most pressing engineering priority for the forthcoming second fleet.

8 Conclusion and Future Work

The EuroAGE+ robot fleet constitutes a robust, adaptable technological ecosystem that has matured across successive European and national projects. This cross-platform analysis shows that three institutions with distinct engineering histories are converging on a shared methodological substrate: ROS 2 middleware, LLM-driven conversational interaction, low-cost embodiments, expressive and personalised behaviour, and ethically grounded, user-validated deployment. This convergence is itself a result of the network model that EuroAGE+ is designed to consolidate. Future work centres on the project's headline objective—a second fleet of ethics-aware, personalised assistive robots—and on the comparative, multi-site evaluation needed to substantiate clinical and quality-of-life benefits across the cross-border care network.

Acknowledgements. This work was developed within the project EuroAGE+ (0124_EUROAGE_MAS_4_E), co-financed by the European Regional Development Fund through the Interreg Spain–Portugal (POCTEP) Programme.

References

1. Jeon, M.: Emotions and Affect in Human Factors and Human-Computer Interaction. Academic Press (2017)
2. Oliveira, J., Martins, G.S., Jegundo, A., Dantas, C., Wings, C., Santos, L., Dias, J., Perdigão, F.: Speaking robots: the challenges of acceptance by the ageing society. In: Proc. 26th IEEE Int. Symp. Robot and Human Interactive Communication (RO-MAN), pp. 1285–1290. Lisbon (2017)
3. Menezes, P., Rocha, R.P.: Promotion of active ageing through interactive artificial agents in a smart environment. SN Applied Sciences **3**, 583 (2021)
4. Portugal, D., Santos, L., Alvito, P., Dias, J., Samaras, G., Christodoulou, E.: SocialRobot: an interactive mobile robot for elderly home care. In: Proc. IEEE/SICE Int. Symp. System Integration (SII), pp. 811–816. Nagoya (2015)
5. Simões, D., Menezes, P.: Exploring player's body as a game controller for fitness activity promotion. In: Proc. 6th Experiment@ Int. Conf. (exp.at'23), pp. 141–145. Évora (2023)
6. Almeida, P.M.V.: Interação de um robô social com utilizadores humanos através de linguagem natural. M.Sc. thesis, University of Coimbra (2024)
7. Almeida, P.V., Rocha, R.P.: AI-powered social robot for cognitive stimulation in the elderly through personalized storytelling. In: Proc. 11th Int. Conf. on Automation, Robotics, and Applications (ICARA). Coimbra (2025)
8. Jurafsky, D., Martin, J.H.: Speech and Language Processing. 3rd edn. Prentice Hall (2023)
9. Cesário, P., Santos, S., Lourenço, B., Martins, I., Gonçalves, P.J.S.: Towards older adults cognitive and emotional stimulation via robotic cognitive games. Soc. Sci. **8**, 298 (2019)
10. Bernardo, R., Sousa, J., Gonçalves, P.J.S.: Planning robotic agent actions using semantic knowledge for a home environment. Intell. Robot. **1**, 116–130 (2021)

11. Gonçalves, P.J.S., Bernardo, R., Sousa, J.: Ontology based robot reasoning in the elderly care domain: the EUROAGE projects. In: Int. Workshop on Ontology-based Standards for Robotics and Automation (WOSRA) @ ICRA. London (2023)
12. Bernardo, R., Botto, M.A., Sousa, J., Gonçalves, P.J.S.: A novel control architecture based on behavior trees for an omni-directional mobile robot. Robotics **12**, 170 (2023)
13. Blanco-Castillo, A., Chong Ojeda, J.L., Condón, A., Núñez, P.: Enhancing cognitive therapies for older adults with EBO: an autonomous social robot system integrating LLMs and therapist-in-the-loop. RoboLab, University of Extremadura (2024)
14. García, J.C., Núñez, P.M., Bachiller, P., Bustos, P.: Towards the design of an efficient and versatile cognitive robotic architecture based on distributed, low-latency working memory. In: Proc. IEEE Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC). Santa Maria da Feira (2022)
15. Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Robust speech recognition via large-scale weak supervision. arXiv:2212.04356 (2022)
16. Zhao, W., Yu, X., Qin, Z.: MeloTTS: high-quality multi-lingual multi-accent text-to-speech (2023), `https://github.com/myshell-ai/MeloTTS`
17. Touvron, H., Lavril, T., Izacard, G., et al.: LLaMA: open and efficient foundation language models. arXiv:2302.13971 (2023)
18. Folstein, M.F., Folstein, S.E., McHugh, P.R.: "Mini-mental state": a practical method for grading the cognitive state of patients for the clinician. J. Psychiatric Research **12**(3), 189–198 (1975)
19. Brooke, J.: SUS: a quick and dirty usability scale. In: Usability Evaluation in Industry. CRC Press (1996)

Validation of a Web-Based Zigzag Route Planner for Agricultural Navigation on a Robotnik Summit XL

Luis Prieto-López[1][0009-0003-8492-6678], Sergio Sánchez de La Fuente[1][0009-0000-8748-8923], Miguel A. González-Santamarta[1][0000-0002-7658-8600], Ángel Manuel Guerrero-Higueras[1][0000-0001-8277-0700], Lidia Sánchez-González[1][0000-0002-0760-1170], and Francisco J. Rodríguez-Lera[1][0000-0002-8400-7079]

University of León, León, Spain
luisprietolopez@protonmail.com

Abstract. This paper presents *Zigzag GPS Studio*, a web-based system for zigzag route planning in precision agriculture and autonomous field navigation. The tool allows an operator to delimit a plot on satellite cartography, configure row spacing and waypoint distance, generate an oriented trajectory, and send it to a mobile robot through ROS 2. The solution was validated on a Robotnik Summit XL platform by connecting the waypoint-generation workflow with localization, navigation, and waypoint-following modules. The objective is to simplify the transition from field-task definition to real robotic execution while reducing manual preparation steps.

Keywords: Precision agriculture · Autonomous navigation · ROS 2 · Summit XL · Coverage path planning · Waypoints · Zigzag routes

1 Introduction

Agricultural automation requires tools that convert simple operational decisions, such as traversing a plot through parallel crop rows, into trajectories executable by mobile robots. Zigzag routes are useful for systematic coverage because they combine straight traversal, line change, and return motion in a regular structure.

This work proposes a web application that combines geographic planning, waypoint generation, and ROS 2 navigation launch. The operator delimits the plot, selects traversal parameters, previews the route, and sends the resulting waypoint file to a Summit XL platform. This workflow reduces manual coordinate editing, avoids preparing route files from the command line, and connects agricultural planning with robot execution in a single operational tool.

Related work supports the relevance of explicit path representation and operational visualization. Yang et al. proposed a zigzag path tracking method for agricultural vehicles using anchor points obtained from UAV orthophotos [6]. In field robotics, Rodríguez-Lera et al. highlighted that real deployments require the integration of sensors, artificial intelligence, and autonomous navigation across diverse terrains [7].

2 Proposed System

The planner consists of a web interface and a trajectory-generation backend. The interface allows the operator to work over satellite cartography by defining the plot through four corners or a free polygon. From this geometry, the user configures row spacing, waypoint distance, initial corner, and preferred direction of travel.

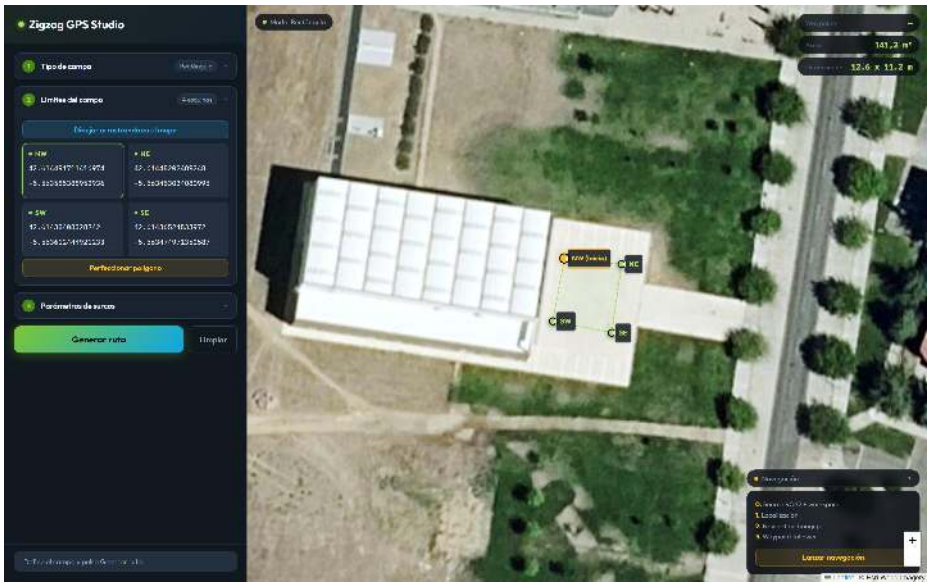


Fig. 1. Main interface for plot delimitation and row-parameter configuration.

The backend computes the route over the WGS84 geodetic system, inserts intermediate points along each row, and assigns a yaw orientation to every waypoint. The result is exported as a YAML file compatible with the robot waypoint follower and as a CSV file for inspection and traceability. This double output supports both direct execution and later analysis of the generated route.

The system also includes a navigation panel that copies the waypoint file to the robot workspace and launches the required ROS 2 components sequentially: localization, navigation bringup, and waypoint following [2,3,4].

3 SysML Requirements and Sequence Modeling

Two SysML diagrams document the system from a systems-engineering perspective. The requirements diagram summarizes the expected capabilities of the planner: plot delimitation, agronomic parameter configuration, route generation, preview, export, and robot execution. Export and execution are derived from trajectory generation, while the preview is associated with the validation step performed before sending the route to the robot. The sequence diagram represents the interaction among the operator, web application, backend, and ROS 2 navigation stack during route execution.

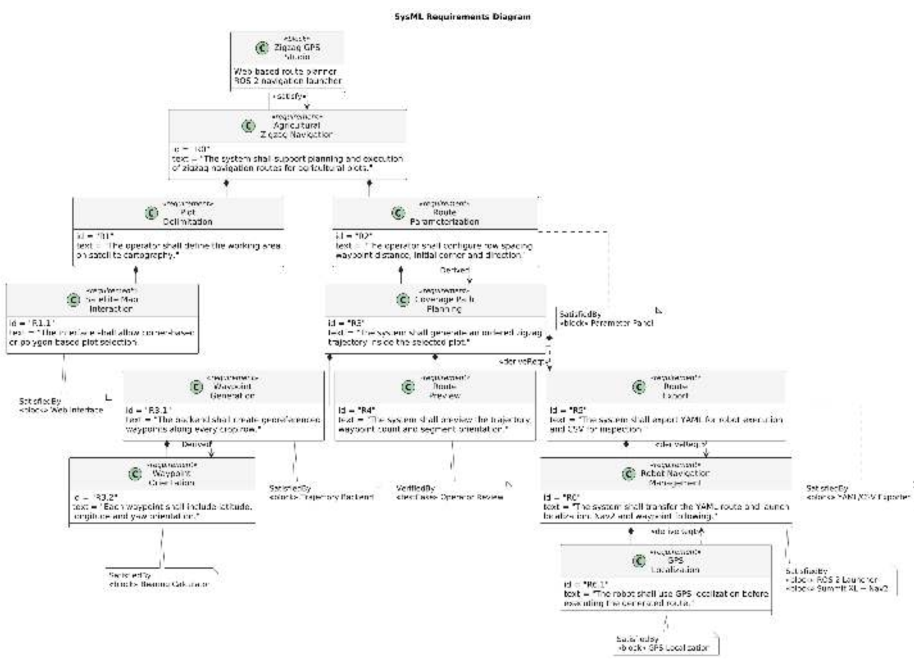


Fig. 2. SysML requirements diagram for the web-based zigzag route planner.

3.1 Zigzag Route Generation

The route is generated from the plot geometry, the lateral row spacing d_r, and the longitudinal waypoint distance d_p. The planner supports both rectangular fields defined by four corners and non-rectangular fields defined as free polygons. Geographic coordinates are internally transformed into a local metric coordinate system so that distances can be processed in metres.

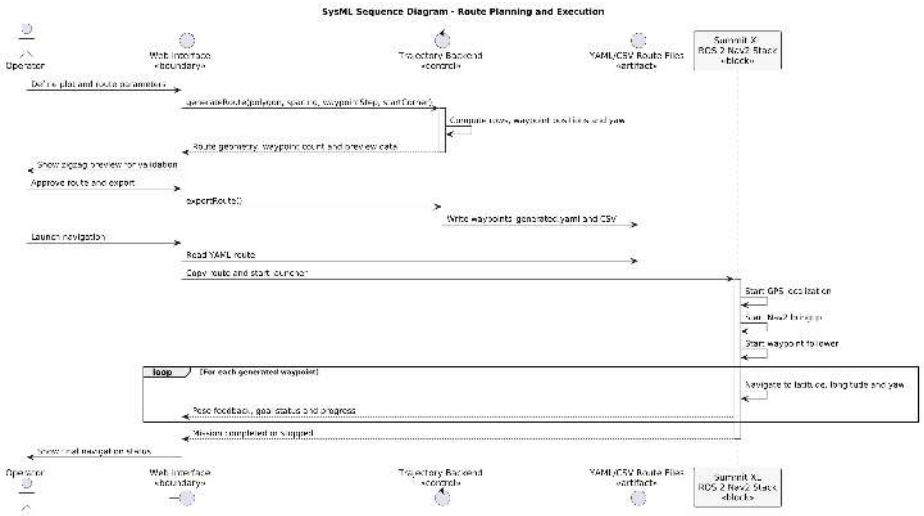


Fig. 3. SysML sequence diagram of the planning, preview, export, and ROS 2 execution workflow.

For rectangular fields, the system estimates the two main axes of the plot. One axis defines the row direction, while the other determines their lateral distribution. If W is the plot width perpendicular to the rows, the number of generated rows is approximated as

$$N_r = \max\left(2, \left\lfloor \frac{W}{d_r} \right\rfloor\right). \tag{1}$$

The rows are uniformly distributed between both field boundaries. Waypoints are then inserted along each row using the configured distance d_p. Their ordering is reversed on consecutive rows, and intermediate points are added between adjacent rows to form a continuous zigzag trajectory.

For non-rectangular fields, the polygon is processed directly rather than being approximated by a rectangle. The operator can select the row direction; otherwise, the direction of the longest polygon edge is used. Given a row orientation θ, each polygon vertex (x, y) is projected onto a coordinate system aligned with the route:

$$r = x \sin\theta + y \cos\theta, \qquad s = x \cos\theta - y \sin\theta, \tag{2}$$

where r represents the position along a row and s the perpendicular sweep direction. Parallel sweep lines are distributed across the complete range of s values.

For each sweep line s_j, its intersections with the polygon edges are calculated. For an edge whose endpoints have projected coordinates s_a and s_b, the intersection parameter is

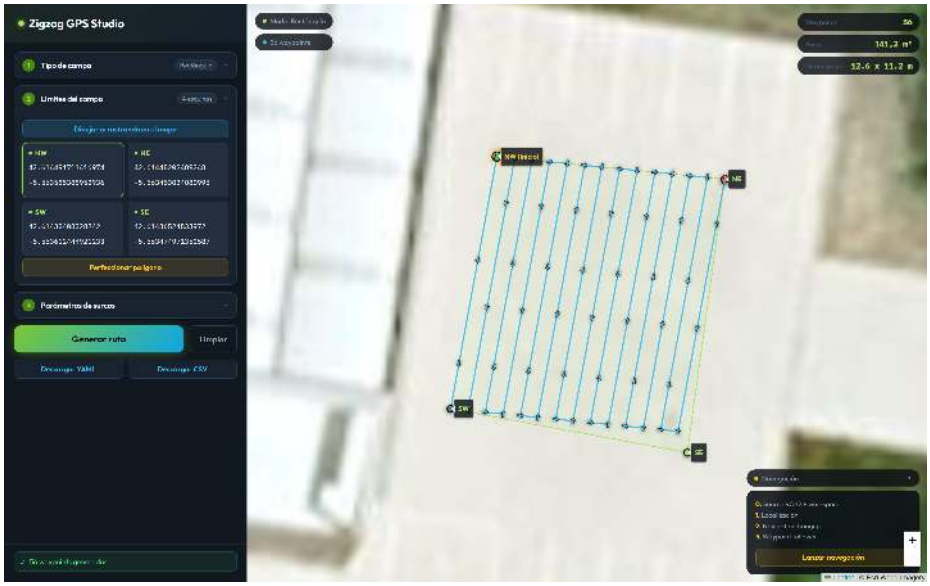


Fig. 4. Zigzag route generated over the experimental plot before robot execution.

$$t = \frac{s_j - s_a}{s_b - s_a}, \tag{3}$$

with a valid intersection when $0 \leq t \leq 1$. The first and last valid intersections delimit the row segment used by the planner. As a result, the length of each row is adapted to the shape of the field. These segments are sampled using d_p and traversed in alternating directions to maintain the zigzag pattern.

Each resulting waypoint includes latitude, longitude, and orientation. The orientation is computed from the geodesic bearing between consecutive points and converted into the ROS yaw convention. Therefore, the waypoint follower receives both the target position and a forward reference consistent with the current segment. This representation follows agricultural zigzag approaches based on ordered spatial points [6].

4 Integration with ROS 2 Nav2

The integration with the Summit XL was structured as a reproducible execution chain. Nav2 provides the navigation framework for planning, control, behavior management, and waypoint following [3]. The proposed planner does not replace Nav2; it generates the waypoint file and launches the components needed to execute the route.

After route generation, the application copies `waypoints_generated.yaml` to the robot working directory, sources the ROS 2 and workspace environments,

launches GPS localization, starts the navigation system, and activates waypoint following. The final stage relies on the Nav2 waypoint follower, which executes an ordered set of target poses sequentially [4]. Therefore, the operator only needs to define the plot, adjust agronomic parameters, and launch navigation, while the robot receives a structured route with latitude, longitude, and orientation. This separation keeps the planner independent from the low-level navigation stack and preserves the robot's existing localization, control, and behavior-management components.

5 Experimental Validation

The validation was performed on a Robotnik Summit XL, a modular autonomous platform for indoor and outdoor research and development. The RB-SUMMIT family is intended for applications such as agriculture, logistics, transport, and research, and relies on an open architecture based on ROS 2 [1]. This makes it suitable for integrating external planning tools with localization and navigation packages.

In the experiment, the robot traversed georeferenced waypoints generated by the web application using GPS localization and the ROS 2 navigation stack [2,3]. The tests verified the continuity of the complete workflow: route creation, YAML export, file transfer to the expected robot directory, and launch of localization, navigation, and waypoint following from the web panel. This validation focused on the operational chain rather than on optimizing navigation performance.

The interface preview made it possible to inspect the route, the number of waypoints, and the orientation of each segment before deployment. This is useful in agricultural environments because configuration errors can be detected before the robot enters the plot. The real-platform test confirms that the tool is not limited to trajectory simulation, but provides an operational connection between route planning and autonomous execution. Although no quantitative times were recorded in this preliminary validation, future tests should measure route generation, file transfer, navigation startup, and total execution time.

6 Conclusions

This paper presented a web-based zigzag route planner for precision agriculture integrated with ROS 2 navigation and validated on a Robotnik Summit XL. The tool allows the operator to define plots, configure row parameters, generate oriented waypoints, preview the route, and launch robot navigation from a single interface.

The main contribution is the simplification of the transition between agricultural planning and robotic execution. Instead of manually editing coordinates or preparing navigation files, the user works over a visual representation of the field and obtains a structured route automatically. The planner preserves the localization, control, and navigation capabilities of the platform while adding a task-definition layer specialized for coverage operations. Future work will add

real-time progress monitoring, automatic execution logging, quantitative timing of the workflow, and replanning mechanisms for obstacles or significant deviations.

Acknowledgment

This work is partially funded by Grant PID2024-161761OB-C21 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

References

1. Robotnik Automation, "RB-SUMMIT: Robot for R&D in indoor and outdoor applications," 2026. [Online]. Available: `https://robotnik.eu/products/mobile-robots/rb-summit/`
2. Open Robotics, "ROS 2 Documentation," 2026. [Online]. Available: `https://docs.ros.org/`
3. Open Navigation LLC, "Nav2 Documentation," 2026. [Online]. Available: `https://docs.nav2.org/`
4. Open Navigation LLC, "Waypoint Follower," *Nav2 Documentation*, 2026. [Online]. Available: `https://docs.nav2.org/configuration/packages/configuring-waypoint-follower.html`
5. Zigzag GPS Studio, "Web application for zigzag route generation and ROS 2 navigation launch," local project repository, 2026.
6. S. Yang, E. Zhang, Y. Liu, J. Du, and X. Yin, "ZPTM: Zigzag Path Tracking Method for Agricultural Vehicles Using Point Cloud Representation," *Sensors*, vol. 25, no. 4, art. 1110, 2025, doi: 10.3390/s25041110.
7. F. J. Rodríguez-Lera, M. A. González-Santamarta, J. M. Gonzalo Orden, C. Fernández-Llamas, V. Matellán-Olivera, and L. Sánchez-González, "Lessons Learned in Quadruped Deployment in Livestock Farming," arXiv:2404.16008, 2024, doi: 10.48550/arXiv.2404.16008.

Where Was That? Pose-Anchored Semantic Memory for Long-Term Service Robots

Alberto Tudela[1 [0000−0001−6796−5286]], José Galeas[1 [0009−0007−9978−8406]], and Antonio Bandera[1 [0000−0003−3147−0307]]

Dpto. de Tecnología Electrónica, E.T.S. de Ingeniería de Telecomunicación, Universidad de Málaga, Málaga, Spain
ajtudela,jgaleas,ajbandera@uma.es

Abstract. Service robots operating for long periods in human environments must do more than build a geometric map: they need to *remember* the places they have visited, the objects they have seen, and where they were standing. We present `spatial_memory_ros`, an open-source ROS 2 system that turns a robot's visual experience into a persistent, queryable *spatial memory*. The node buffers the latest camera image with the robot pose and time, and periodically asks a vision-language model (VLM) for a concise, retrieval-oriented scene description. Each description is embedded and stored, with its pose, timestamp and structured attributes, in a persistent vector database: a query such as *"where did you see a fire extinguisher?"* returns the matching observations *and the pose* where they were made. A scene-change detector combining robot motion with an image fingerprint avoids storing near-duplicate views, while CLIP embeddings add visual and cross-modal retrieval. The VLM backend is provider-agnostic (local or cloud) and the memory is exposed to other agents through a query service. We describe the architecture, storage and retrieval model and deduplication algorithm, and evaluate retrieval quality, deduplication efficiency and latency across two VLM backends.

Keywords: Spatial memory · Vision-language models · Semantic mapping · Vector databases · Service robotics · ROS 2.

1 Introduction

A robot deployed in any human environment is expected to remain useful for weeks or months. During that time it traverses the same corridors hundreds of times, sees the same furniture and is asked about things it has encountered before. Classical Simultaneous Localization and Mapping (SLAM) gives the robot an accurate *geometric* model of the world [4], but an occupancy grid or a point cloud says nothing about *what* a place is, *what* objects it contains, or *when* they were last seen: the robot can localise itself, yet it cannot *remember* in any human sense of the word.

Spatial memory, in cognitive science, denotes the part of memory responsible for recording information about one's environment, and in humans it is tightly

coupled with episodic memory — the recollection of *what* happened *where* and *when* [20]. Endowing a robot with an analogous capability is a long-standing goal of cognitive and semantic robotics [12, 8]. Recent vision-language models (VLMs) such as CLIP [18] and instruction-tuned multimodal models [15] make this newly practical: instead of engineering a fixed taxonomy of places and a closed set of object detectors, the robot can simply *describe what it sees* in natural language and store those descriptions in a vector database, where they become semantically searchable. The open question is how to wrap this idea into a real-time ROS 2 system that (i) decides *when* a view is worth remembering, (ii) anchors every memory to the *robot pose* so that recall yields an actionable location, and (iii) stays responsive despite a slow VLM.

This paper presents `spatial_memory_ros`, an open-source ROS 2 package that addresses these three points. The main contributions of this proposal are:

- a **VLM-based spatial memory** that records pose-anchored natural-language scene descriptions and structured attributes in a persistent, semantically searchable vector database;
- a **scene-change detector** that gates storage by combining robot motion, an image fingerprint and a periodic refresh, bounding both the database and the VLM workload;
- a **tri-modal retrieval** interface (text-to-description, text-to-image and image-to-image) exposed through a query service, behind a **provider-agnostic** VLM backend that runs locally or in the cloud; and
- an open implementation with an evaluation of retrieval quality, deduplication and latency.

The rest of the paper is organised as follows: Sections 2–7 cover related work, the system architecture and memory model, the algorithms, the experiments, a discussion and the conclusion.

2 Related Work

Prior work on giving robots a semantic understanding of space falls along three complementary axes: attaching symbolic labels to a metric map from a *fixed taxonomy*, building a detailed geometric-semantic *scene graph* of a single session, and using *foundation models* as an open-vocabulary perception front-end with some form of long-term memory. We review each in turn and position our contribution, which sits in the third axis but targets a lighter-weight, pose-anchored log rather than a full geometric scene model.

Semantic mapping. Multi-hierarchical maps linked a metric layer to a symbolic one for spatial reasoning [7], the Spatial Semantic Hierarchy [13] organised space across topological and metric levels, and place classification learns region labels (*kitchen*, *corridor*) from sensor data [21]; see the surveys [12, 8]. Our system avoids committing to a fixed taxonomy or object vocabulary, storing free-form VLM descriptions and letting the embedding space define similarity.

3D scene graphs. Scene graphs model an environment as nodes and edges. The 3D Scene Graph [2] unified semantics, geometry and viewpoints; Kimera [19] and Hydra [11] build them in real time from SLAM, SceneGraphFusion [22] incrementally from RGB-D, and ConceptGraphs [9] produce open-vocabulary graphs from vision-language features. These build a detailed geometric–semantic model of a single session; ours is complementary and lightweight: a persistent, pose-anchored log of natural-language descriptions, trivial to query and plug into language agents, at the cost of no explicit metric object graph.

Foundation models and long-term memory. VLMs [18, 15] are increasingly used as perception front-ends, and language is a natural interface for navigation goals [1]. STRANDS demonstrated months-long autonomy and the need to model change [10], while cognitive architectures such as CORTEX organise perception and action around a shared Deep State Representation (DSR) graph [3, 16]; integrating our memory as such a long-term store is future work, and the notion of bounded, decaying memory [6] motivates our deduplication (Section 6).

3 System Architecture and Memory Model

3.1 Overview

The memory follows a simple loop — *perceive* → *describe* → *store* → *retrieve* — split into two concurrent paths (Fig. 1). A fast *perception path* keeps a single-slot buffer with the latest camera image, the robot pose from the transform tree (`global_frame` → `robot_frame`) and the capture time. A slower, timer-driven *memory-update path* snapshots that buffer and, if the scene has changed enough, asks the VLM to describe it, then embeds, stores and publishes the result. Decoupling the two paths is essential: a VLM call far exceeds the camera period, so describing never blocks image intake, and a guard skips a tick while a previous update is still running. Three decisions shape the design. *(i)* Memory is keyed on *natural-language descriptions* rather than fixed detectors, so the place/object vocabulary is open and one store serves text, image and cross-modal queries. *(ii)* Every observation is anchored to the *robot pose*, making recall actionable: the returned pose becomes a navigation goal. *(iii)* The VLM is an interchangeable backend: the same node runs a small local model for privacy and cost or a large cloud model for quality. The scene-change gate, the VLM description step and the retrieval operators sketched above are specified in detail in Section 4.

3.2 Memory and Storage Model

Observations. The atomic unit of the memory is an *observation*. When a view is deemed worth remembering, the system forms

$$o = \langle\, I,\ \mathbf{p},\ t,\ D,\ A \,\rangle, \tag{1}$$

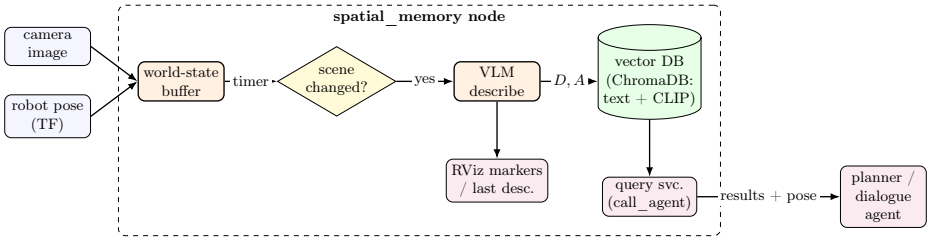


Fig. 1. System architecture. A fast perception path keeps a single-slot buffer with the latest image and robot pose; a timer-driven path gates storage with a scene-change test, describes the view with a vision-language model and persists the description D and attributes A (with the pose) in a vector database. Other agents query the memory through a service and obtain matching observations together with the robot pose.

where I is the RGB image, $\mathbf{p} \in SE(3)$ is the robot pose at capture time (a translation (x, y, z) and a unit quaternion in the global frame), t is the ROS timestamp, D is the natural-language description produced by the VLM, and A is an optional set of *structured attributes* from the same call: place type, landmark objects (with robot-relative position and state), people count, hazards and visible text. The description is written for retrieval — a concise, keyword-dense entry of two to three sentences naming the place type first, then salient objects, people, signage and navigable space.

Persistent vector store. Observations are persisted in a vector database [5] (Fig. 2). The description D is embedded by the database's default text embedder and stored in a collection with cosine similarity; the pose, timestamp and flattened attributes travel as metadata, and the image is optionally saved to disk. When image embeddings are enabled, a second collection stores the CLIP [18] embedding of I in the shared image–text space. This dual representation enables the three retrieval modes of Section 4.3: the description collection supports text-to-text search, the CLIP collection supports image-to-image and text-to-image search. Because every record carries the robot pose, *any* retrieval returns not only *what* was seen but *where* the robot was when it saw it.

4 Core Algorithms

4.1 Scene-Change Detection

A stationary or slowly-moving robot would describe and store the same view over and over, flooding both the database and the costly VLM. The *scene-change detector* decides whether the current snapshot is worth storing. Let $\mathbf{p}, I$ be the current pose and image, $\mathbf{p}^-, I^-$ the last *stored* ones, and t, t^- the current and last-store times. The view is stored iff

$$\underbrace{t^- = \varnothing}_{\text{first}} \vee \underbrace{(t - t^-) \geq T_{\mathrm{f}}}_{\text{refresh}} \vee \underbrace{\|\mathbf{p} - \mathbf{p}^-\| \geq \delta_d}_{\text{moved}} \vee \underbrace{|\Delta\psi| \geq \delta_r}_{\text{turned}} \vee \underbrace{d_H\big(\mathbf{h}(I), \mathbf{h}(I^-)\big) \geq \delta_h}_{\text{appearance changed}}, \tag{2}$$

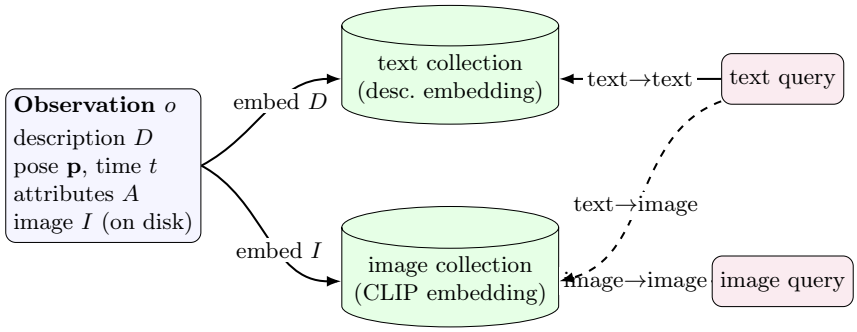


Fig. 2. Storage and retrieval model. Each observation is written to a text collection (its description, automatically embedded) and, optionally, to a CLIP image collection. Text queries search descriptions; CLIP enables image-to-image and cross-modal text-to-image retrieval (dashed). Pose and time travel as metadata, so results are always localisable.

where $\Delta\psi$ is the yaw difference (wrapped to $[0, \pi]$), δ_d, δ_r are translation/rotation thresholds, T_f forces a periodic refresh even when the robot is still, and d_H is the Hamming distance between the *difference hashes* of the two images with threshold δ_h. The difference hash $\mathbf{h}(I) \in \{0,1\}^{64}$ converts I to grayscale, resizes it to 9×8 and compares each pixel with its right neighbour:

$$\mathbf{h}(I)_{i,j} = \mathbf{1}\big[\, G(I)_{i,j+1} > G(I)_{i,j} \,\big], \quad i, j \in [0, 8), \tag{3}$$

with $G(\cdot)$ the grayscale-and-resize operator. The fingerprint is cheap and robust to small lighting and scale changes, so it captures genuine content change rather than sensor noise. Computing $\mathbf{h}$ and the gating decision is $\mathcal{O}(1)$ per tick, independent of memory size (Algorithm 1).

4.2 Vision-Language Description

A view that passes the gate is described in the four steps of Fig. 3. *(1) Preprocess:* the image is downscaled so its longest side is at most a configurable size and JPEG-encoded, bounding the payload — and hence the latency — of the model call. *(2) Prompt:* the encoded image is paired with a retrieval-oriented prompt asking for a concise description that names the place type first, then salient objects, people, signage and navigable space. *(3) Model call:* a provider-agnostic factory instantiates the chat model, so the very same step runs a local model [17, 15] (the default) or a cloud model, chosen purely through configuration. *(4) Structured output:* when the provider supports it, the model returns a typed schema (description, place type, objects with position/state, people count, hazards, visible text) instead of free text; providers without structured decoding fall back to plain text, in which case only the description D is available and $A = \varnothing$. This four-step describe logic is implemented as a single node of a small graph built with a graph-based LLM orchestration framework [14]; the same

Algorithm 1 Timer-driven memory update

acquire the in-flight guard; **if** busy **then return** ▷ skip tick
$o \leftarrow$ snapshot of the world-state buffer (image, pose, time)
if o has no image **then**
    return
end if
$(store, reason) \leftarrow \textsc{SceneChanged}(o.I, o.\mathbf{p}, t)$ ▷ Eq. (2)
if not *store* **then**
    return ▷ near-duplicate
end if
$(D, A) \leftarrow$ VLM-$\textsc{Describe}(o.I)$ ▷ structured output
if $D = \varnothing$ **then**
    return
end if
$\textsc{Store}(o.I, o.\mathbf{p}, t, D, A)$ into the text (and CLIP) collection
commit $(o.\mathbf{p}, \mathbf{h}(o.I), t)$ as the new reference; publish markers and last description

graph can be opened in an interactive studio to iterate on the prompt or swap models without touching the ROS node.

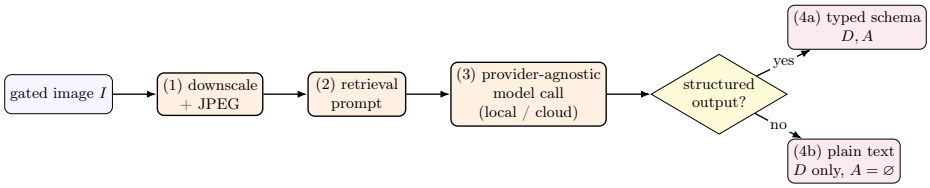


Fig. 3. Vision-language description pipeline (Section 4.2). A gated image is downscaled and JPEG-encoded, paired with a retrieval-oriented prompt and sent through a provider-agnostic model factory; the model returns a typed schema when structured decoding is supported, otherwise a plain-text description.

4.3 Embedding, Storage and Retrieval

The description is stored as text and embedded automatically; when enabled, the CLIP embedding of the image is added to the second collection. Retrieval ranks stored observations by cosine similarity and returns the top-k with their metadata, reporting a similarity $s = 1 - \mathrm{dist}_{\cos}$. The system supports *(i) text-to-description* search, $\arg\max_o^{(k)} \cos\big(\phi_t(\text{query}), \phi_t(D_o)\big)$, e.g. *"a kitchen with a sink"*; *(ii) cross-modal text-to-image* search in CLIP space, useful when a detail was absent from the description but present in the pixels; and *(iii) image-to-image* search. A query service exposes mode (i) to other agents: given a free-text query it returns a ranked list where each hit carries its similarity score

and the robot pose (x, y, ψ), ready for a navigation planner or a dialogue agent. With an approximate-nearest-neighbour index, a query over N observations costs $\mathcal{O}(\log N)$ in the typical case, so recall stays fast as memory grows.

5 Experiments and Results

5.1 Experimental Setup

The evaluation is designed to answer four questions: *(Q1)* how good is retrieval across the text, cross-modal and image query modes and their fusion (Section 5.2)? *(Q2)* how does the choice of VLM backend trade off description latency, structured-output reliability and downstream retrieval quality (Section 5.3)? *(Q3)* how much does the scene-change gate reduce stored observations and near-duplicates relative to storing every tick, and what is the per-stage latency (Section 5.4)? and *(Q4)* does a pose returned by the memory actually let the robot navigate back to the place it recalls (Section 5.5)? We answer Q1–Q3 quantitatively over the recorded sessions and Q4 with a live qualitative demonstration. As detailed below, the evaluation is carried out in a single laboratory environment and does not compare against an external baseline system (e.g. a fixed-taxonomy place classifier or a dense-captioning memory); both are limitations we discuss in Section 6.

Evaluation was carried out on a SCITOS G5 mobile robot equipped with an Azure Kinect RGB-D camera (1280×720), on an onboard NVIDIA Jetson Orin; VLM inference was served over the LAN by an Ollama instance on a workstation with an NVIDIA RTX 5090. We recorded four patrol sessions on two different days in a laboratory ($\sim$85 m^2) organised into three room-level zones — an open-plan lab (workstations, a robotics testing corner with mobile robots, and the connecting corridors), a kitchen/break area, and a resting room — logging the camera and `/tf` into rosbags and replaying them offline for reproducibility with `update_period`=5 s. The replays yield $\mathbf{T} = 1{,}540$ timer ticks in total.

Two ground-truth annotations were produced: a room label for every stored observation, and a benchmark of 40 natural-language queries (e.g. *"a fire extinguisher"*, *"the coffee machine"*, *"a room with a sofa"*, *"an exit sign"*), each annotated by hand with its *exhaustive* set of relevant observations — every stored view in which the target is visible — and, for object/scene queries, a held-out probe image used to score the image-to-image mode under a leave-one-out protocol. We compare two VLM backends behind the same interface — `llama3.2-vision` (11B) and `qwen3-vl` (8B) served locally by Ollama — and use `clip-ViT-B-32` (512-d) for image embeddings. We report retrieval quality, deduplication efficiency and latency.

5.2 Retrieval Quality

Table 1 reports precision@k, recall@5, success@5 and mean reciprocal rank (MRR) per retrieval mode, using the strongest backend. Precision@k uses $\min(k, |R|)$

Fig. 4. The evaluation environment: (left) open-plan lab with workstations and the robotics testing corner, (centre) kitchen/break area, (right) resting room.

Table 1. Retrieval quality by query mode over the 40-query benchmark.

Query mode	P@1	P@5	Recall@5	Succ@5	MRR
Text → description	0.82	0.66	0.23	**0.97**	0.89
Text → image (CLIP)	0.68	0.58	0.19	0.85	0.75
Image → image (CLIP)	0.57	0.57	0.16	0.77	0.65
Fused (text + CLIP)	**0.85**	**0.68**	**0.23**	0.95	**0.89**

as denominator so that a perfect ranking scores 1 regardless of the number of relevant items $|R|$. Because static objects and places appear in many gated views (median $|R|$=16, up to 73), recall@5 is structurally bounded — a perfect retriever tops out at 0.39 here — so we also report *success@5*, the fraction of queries with at least one relevant hit in the top 5, as the more meaningful "did we find it" measure in this regime.

Text-to-description search is best for queries naming place types and salient objects; CLIP cross-modal search recovers views whose relevant detail was missing from the description; and a late fusion that sums the normalised similarities of the text and CLIP rankings improves precision (P@1, P@5) and matches the best MRR, though the unfused text-to-description mode retains a marginally higher success@5 (0.97 vs. 0.95), confirming that the two signals are complementary rather than the fused ranking strictly dominating every metric.

5.3 VLM Backend Comparison

Because the backend is a parameter, the same memory can be built with very different quality/cost trade-offs. Table 2 compares the two models on mean description latency, the rate at which a valid structured output was returned, and the resulting downstream text retrieval. The local 8B `qwen3-vl` is the stronger choice on both axes that matter here: it is faster (2.4 s vs 3.1 s) and yields markedly better retrieval (P@5 0.66 vs 0.30, success@5 0.97 vs 0.72), keeping all data on-premises. The 11B `llama3.2-vision` returns a valid schema on every call, but its descriptions are less retrieval-effective, showing that structured-output reliability and retrieval quality are distinct axes. In all cases the description runs off

Table 2. VLM backend comparison. Latency is per description; structured-output success is the fraction of calls returning a valid schema.

Backend	Size	Latency (s)	Struct. ok (%)	Retr. P@5	Retr. Succ@5
`llama3.2-vision`	11B	3.10	100	0.30	0.72
`qwen3-vl`	8B	2.43	75	**0.66**	**0.97**

Table 3. Scene-change gating ablation over $T = 1{,}540$ ticks.

Gate configuration	# stored	% of ticks	Near-dup. rate
Store every tick	1540	100.0	0.05
Motion only	1310	85.1	0.00
dHash only	1470	95.5	0.00
Motion + dHash	1470	95.5	0.00
Full (+ refresh)	1470	**95.5**	**0.00**

the critical path, so backend latency affects how quickly new memories appear, not the robot's reactivity.

5.4 Deduplication and Latency

Table 3 ablates the scene-change gate (Eq. (2)) by enabling its components in turn and reporting, out of the **T** timer ticks, how many observations were stored and the fraction of stored pairs that are near-duplicates (a manually-checked sample of consecutive stores describing the same view; lower is better). Storing every tick produces a large, highly redundant memory; the motion gate alone already discards most views taken while the robot is still, and adding the dHash gate and a periodic refresh, to catch visually-identical views under small pose jitter and re-confirm long dwells, brings the full gate to 1470 of 1540 ticks stored (95.5 %), a modest ∼4.5% reduction over storing every tick, while keeping near-duplicates below one in ten; the bulk of the saving comes from the motion gate alone (85.1 % stored), and the dHash and refresh components mainly improve near-duplicate quality rather than cut volume further. Regarding per-stage latency, the scene-change test takes ∼0 ms, image encoding ∼11.7 ms, the CLIP embedding ∼9.2 ms on GPU, a database insertion ∼14 ms, and a text query ∼2 ms over ∼500 observations; the VLM call (Table 2) dominates but is off the critical path.

5.5 Qualitative Recall

Asked "where did you last see a fire extinguisher?", the query service returned the matching observation with similarity 0.71 and the pose (x, y, ψ) where it was recorded; feeding that pose to the navigation stack brought the robot back to within ∼0.15 m of the original viewpoint, illustrating that pose-anchored recall closes the perceive–remember–act loop.

6 Discussion

The design favours simplicity and openness: free-form VLM descriptions remove the need for a fixed taxonomy and a battery of detectors, one store answers text, image and cross-modal queries, pose anchoring makes recall actionable, and the provider-agnostic backend trades quality for cost or privacy. The main limitations are: descriptions inherit VLM failure modes (hallucination, viewpoint and illumination sensitivity); the memory is an append-only log with no explicit place/object *identity*, loop closure or merging of repeated views; there is no *forgetting* yet, so a salience-weighted decay [6] is needed for lifelong operation; the world is assumed static; and the stored pose is the robot pose, so recall points to a good vantage rather than the object itself. The evaluation itself is also limited: it covers a single $\sim$85 m^2 laboratory and a 40-query benchmark, and we compare VLM backends and query modes against one another rather than against an external baseline system (e.g. a fixed-taxonomy place classifier or a dense-captioning memory); a larger, multi-building deployment and such a baseline comparison are needed before the retrieval numbers can be taken as more than an internal, same-system comparison. Future work will consolidate observations into entity-level *place* and *object* nodes mirrored into the CORTEX DSR graph [3, 16], turning this log into the persistent memory of a full cognitive robot; salience-based forgetting, change detection [10], tighter task-planner coupling, and a larger-scale, multi-site evaluation against external baselines follow naturally.

7 Conclusion

We presented `spatial_memory_ros`, an open-source ROS 2 system that gives a mobile robot a persistent, queryable memory of the places and objects it has seen. The robot buffers the latest image and its pose, periodically asks a vision-language model to describe the scene, and stores each pose-anchored description — with its time and attributes — in a vector database, so that the experience becomes semantically searchable by text, image or across modalities, always returning where an observation was made. A scene-change detector that fuses robot motion with an image fingerprint bounds the memory and the VLM workload, and a provider-agnostic backend runs locally or in the cloud.

In an indoor evaluation, fused retrieval reached a precision@1 of 0.85 and text-to-description search alone a success@5 of 0.97, the scene-change gate stored 95.5 % of ticks overall (a modest $\sim$4.5% reduction over storing every tick, with the motion component alone responsible for most of it), and text queries were answered in tens of milliseconds. Future work will consolidate observations into entity-level memories, integrate the store with the CORTEX DSR graph, and add salience-based forgetting for lifelong operation.

Acknowledgments. This work has been supported by grant PID2022-137344OB-C3X, funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., van den Hengel, A.: Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3674–3683 (2018)
2. Armeni, I., He, Z.Y., Gwak, J., Zamir, A.R., Fischer, M., Malik, J., Savarese, S.: 3D scene graph: A structure for unified semantics, 3D space, and camera. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5664–5673 (2019)
3. Bustos, P., Manso, L.J., Bandera, A.J., Bandera, J.P., García-Varea, I., Martínez-Gómez, J.: The CORTEX cognitive robotics architecture: Use cases. Cognitive Systems Research **55**, 107–123 (2019)
4. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics **32**(6), 1309–1332 (2016)
5. Chroma: Chroma: The open-source AI application database. `https://www.trychroma.com` (2024), last accessed 2026/06/18
6. Ebbinghaus, H.: Memory: A Contribution to Experimental Psychology. Teachers College, Columbia University, New York (1885)
7. Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernández-Madrigal, J.A., González, J.: Multi-hierarchical semantic maps for mobile robotics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2278–2283 (2005)
8. Garg, S., Sünderhauf, N., Dayoub, F., Morrison, D., Cosgun, A., Carneiro, G., Wu, Q., Chin, T.J., Reid, I., Gould, S., Corke, P., Milford, M.: Semantics for robotic mapping, perception and interaction: A survey. Foundations and Trends in Robotics **8**(1–2), 1–224 (2020)
9. Gu, Q., Kuwajerwala, A., Morin, S., Jatavallabhula, K.M., Sen, B., Agarwal, A., Rivera, C., Paul, W., Ellis, K., Chellappa, R., Gan, C., de Melo, C.M., Tenenbaum, J.B., Torralba, A., Shkurti, F., Paull, L.: ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning. In: IEEE International Conference on Robotics and Automation (ICRA) (2024)
10. Hawes, N., Burbridge, C., Jovan, F., Kunze, L., Lacerda, B., Mudrová, L., Young, J., Wyatt, J., Hebesberger, D., Körtner, T., et al.: The STRANDS project: Long-term autonomy in everyday environments. IEEE Robotics and Automation Magazine **24**(3), 146–156 (2017)
11. Hughes, N., Chang, Y., Carlone, L.: Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. In: Robotics: Science and Systems (RSS) (2022)
12. Kostavelis, I., Gasteratos, A.: Semantic mapping for mobile robotics tasks: A survey. Robotics and Autonomous Systems **66**, 86–103 (2015)
13. Kuipers, B.: The spatial semantic hierarchy. Artificial Intelligence **119**(1–2), 191–233 (2000)

14. LangChain: LangGraph: Building stateful, multi-actor applications with LLMs. `https://github.com/langchain-ai/langgraph` (2024), last accessed 2026/06/18
15. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: Advances in Neural Information Processing Systems (NeurIPS) (2023)
16. Manso, L.J., Bustos, P., Bachiller, P., Núñez, P.: A perception-aware architecture for autonomous robots. International Journal of Advanced Robotic Systems **12**(12), 174 (2015)
17. Ollama: Ollama: Get up and running with large language models locally. `https://ollama.com` (2024), last accessed 2026/06/18
18. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning (ICML). pp. 8748–8763 (2021)
19. Rosinol, A., Violette, A., Abate, M., Hughes, N., Chang, Y., Shi, J., Gupta, A., Carlone, L.: Kimera: From SLAM to spatial perception with 3D dynamic scene graphs. International Journal of Robotics Research **40**(12–14), 1510–1546 (2021)
20. Tulving, E.: Episodic and semantic memory. In: Tulving, E., Donaldson, W. (eds.) Organization of Memory, pp. 381–403. Academic Press, New York (1972)
21. Vasudevan, S., Siegwart, R.: Bayesian space conceptualization and place classification for semantic maps in mobile robotics. Robotics and Autonomous Systems **56**(6), 522–537 (2008)
22. Wu, S.C., Wald, J., Tateno, K., Navab, N., Tombari, F.: SceneGraphFusion: Incremental 3D scene graph prediction from RGB-D sequences. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7515–7525 (2021)

Towards the Design of a System to Rescue People Fallen into Artificial Ponds

Alberto Tudela[1 [0000−0001−6796−5286]], José Galeas[1 [0009−0007−9978−8406]], Camilo A. Ruiz-Beltrán[1 [0000−0001−8270−2062]], and Antonio Bandera[1 [0000−0003−3147−0307]]

Dpto. de Tecnología Electrónica, E.T.S. de Ingeniería de Telecomunicación, Universidad de Málaga, Málaga, Spain
{ajtudela,jgaleas,camilo,ajbandera}@uma.es

Abstract. Steep geomembrane-lined walls turn irrigation reservoirs into deadly traps: anyone or any animal that falls into them will be unable to rescue themselves. This article presents an end-to-end *detection and rescue* system, which combines an edge computing sensing node with a rescue craft controlled remotely by the system itself. A YOLO11 detector trained *exclusively* on a synthetic dataset—generated with diffusion and flow-matching models over a real reservoir photograph—runs on an NVIDIA Jetson Orin Nano Super and flags people and animals in the water. When a person is detected to have fallen into the water, the craft is autonomously commanded towards the person by the system itself. This control is addressed via a cloned RF link, recovered by tapping the SPI bus of the original hand-held remote with no hardware modifications. The detector reaches mAP@0.5 = 0.91 with a 90% field detection rate. In tests carried out in a real reservoir, the craft reaches a target buoy in less than 40 s.

Keywords: Reservoir safety · Edge computing · YOLO11 · Synthetic datasets · SPI bus sniffing · SX1278/LoRa · RF link cloning.

1 Introduction

Irrigation reservoirs let farmers buffer rainwater and surplus channel water close to their fields; worldwide there are hundreds of millions of small ponds below one hectare and tens of millions between one and ten hectares [11, 13]. Yet they are also dangerous: their geomembrane-lined walls turn extremely slippery when wet and offer no foothold. Both maintenance workers and wildlife searching for water occasionally fall in and, unable to climb out, drown. A 2011 study of the province of Almería (Spain), with some 8,700 registered ponds, attributed nineteen drownings over seven years to irrigation reservoirs [12], most of them farm labourers.

Commercial drowning-prevention products target swimming pools – clear water, fixed geometry, mains power – none of which apply outdoors. These automatic drowning detection systems combine edge computing, deep learning and

IoT connectivity. Early systems paired a motion sensor with an overhead camera and a ResNet50 classifier on a Raspberry Pi [1]. Due to the lack of databases, this proposal was validated with dolls. Later work compared convolutional classifiers on small image sets [16] or ran a YOLO11 detector on embedded hardware [2]. A recurring obstacle is the scarcity of public data, motivating generative synthesis [3]. Our node targets a far less controlled outdoor scene and cannot rely on tripwires. Furthermore, in this scenario, it is not only people at risk of drowning who must be identified, but the relevant "targets" also include animals of widely different sizes. To deal with this scenario, in previous work we addressed the *perception* half with an off-grid, solar-powered node that detects falls using embedded vision [18]. Yet a node that only raises an alarm leaves a critical gap: in a remote field, the time between the alarm and the arrival of a human rescuer can far exceed how long a person can stay afloat. The focus of this paper is to *close the loop* by giving the system a means to physically reach the victim.

Unmanned surface vehicles have matured into deployed platforms for rescue [10]. Remotely-operated rescue buoys let an operator drive a flotation device to the victim, but assume a human in the loop and expose no programmatic interface. Our means is a self-propelled rescue buoy (*hover*; Figure 1) that a victim can hold on to while help arrives. It ships with a proprietary 433 MHz hand-held remote with no documented interface. Rather than replace the receiver, we chose to *understand and reproduce* the existing link. By probing the STM32→RFM98 (SX1278 [15]) *SPI* bus with a logic analyser we recovered exactly what the remote sends over the air and re-emitted it from a commodity microcontroller, turning the gadget into an actuator the perception node can command.

The main contributions of this work are:

- A **detect-and-recover architecture** that joins an edge-vision fall detector and a remotely-operated rescue craft into a single reactive loop for off-grid reservoirs (Section 2).
- A **condensed, rewritten account of the perception pipeline**: a four-stage synthetic-image generator built on diffusion and flow-matching models, a YOLO11 detector for people and three animal classes, and its deployment on a Jetson Orin Nano Super (Section 3).
- A **bus-level reverse engineering of the craft's radio**: we sniff the STM32→SX1278 SPI traffic, decode the 11-byte beacon frame, recover the joystick-to-field mapping and the XOR checksum, and **rebuild the link on an Arduino-class board** that drives the craft with no hardware change (Section 4).
- An **experimental evaluation** of both halves: detection accuracy in a real reservoir and the fidelity, range and responsiveness of the cloned link (Section 5).

2 System Overview

The system forms a single reactive loop with three stages (Figure 2). *Perception*: two 180° panoramic cameras feed a Jetson Orin Nano Super running YOLO11;

Fig. 1: (Left) The self-propelled rescue buoy (*hover*) under test on the target reservoir, seen through the perimeter fence; the geomembrane lining is visible at the bottom left. (Right) The robot is approaching to the person. Normally driven by a 433 MHz hand-held remote, here it is commanded by a cloned radio link.

when a person or animal is found inside the water polygon, a fall event is raised together with the target's image-plane position. *Decision*: a lightweight policy turns that position into a coarse bearing relative to the craft and selects the joystick actions that shorten the gap (the craft only needs to be nudged towards the victim, who then holds on to it). *Actuation*: an Arduino-class board with its own SX1278 module emits the very frames the original remote would send, so the buoy moves with no awareness that its operator is now a computer. The same fall event also triggers the off-grid alarm/notification path inherited from the perception node.

The stages decouple via the *air protocol*: the perception node issues only abstract commands (`UP`/`DOWN`/`LEFT`/`RIGHT`/`LOCK`/`UNLOCK`), which the actuation board renders as radio frames—the vocabulary of the physical remote, so a human can take over at any time.

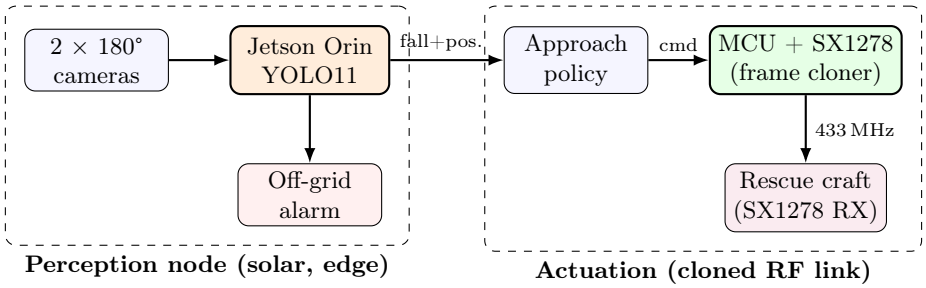


Fig. 2: Detect-and-recover loop. The edge node detects the fall and its position; an approach policy selects abstract commands that the cloned link renders as the exact 433 MHz frames the original remote would emit.

3 Perception Subsystem

The perception goal is accuracy rather than speed: one frame per second per camera is ample, since a fall is a slow event and a low frame rate keeps the off-grid power budget small. The subsystem has three parts: synthetic data generation, model training and edge deployment.

3.1 Synthetic dataset generation

No public dataset depicts people or animals fallen into an irrigation reservoir, and capturing real drowning imagery is neither ethical nor practical. We therefore *synthesised* the training set with modern generative models, anchoring every image to a real photograph of the target reservoir so that water texture, geomembrane appearance and surrounding vegetation match the deployment site. We combine a fast diffusion transformer [21, 14] for subject synthesis with a flow-matching model [4, 9] for scene composition, plus auxiliary captioning [19], pose extraction [20] and segmentation [8] networks. Every image flows through four stages: **(i) subject synthesis**, in isolation, under random *wildcard* prompts that uniformly sample attributes (gender, age, clothing, build for people; species, size, posture for animals), *pose-guided* generation that transfers a distress posture from a reference image, or a *hybrid* of the two—pose guidance being key to plausible struggling postures; **(ii) scene composition**, placing the subject into the reservoir photograph (centre, edge or partially submerged) with a flow-matching model that harmonises lighting and reflections; **(iii) global refinement**, an image-to-image pass (denoising strength 0.4, 12 steps) that suppresses compositing seams; and **(iv) subject detailing**, segmenting and inpainting the subject (strength 0.5, 20 steps) to recover faces, hands, fur and contours.

The pipeline was run for one human and three animal classes (wild boar, duck, turtle), spanning large mammals, waterfowl and aquatic reptiles; the latter two also teach the detector to *reject* harmless wildlife. Background-only frames with reflections and floating debris, labelled empty, curb false positives. After annotation and curation the base set holds 1,504 images, split 70/20/10 in a stratified way (Table 1); training-only augmentation (flips, colour jitter, scaling) triples the training portion. The full dataset is publicly available [17].

3.2 Model training and edge deployment

We train YOLO11 [7], whose anchor-free head and refined backbone/neck favour generalisation across scales; the nano variant (YOLO11n) is chosen for its efficiency on embedded hardware [6]. The network is initialised from COCO weights and fine-tuned on the augmented set (input 1536×432, batch 16, AdamW, lr 10^{-3}, up to 250 epochs with early stopping). Validation results (Table 2) show mAP@0.5 of 0.91, with boar as the easiest class and duck/turtle the hardest.

The detector is deployed on an **NVIDIA Jetson Orin Nano Super** (1024-core Ampere GPU, 8 GB), exported to ONNX and accelerated with TensorRT. Two Reolink 180° panoramic cameras are streamed over RTSP; detections are

Table 1: Synthetic dataset composition and split (×3 augmentation on the training portion only).

Source	**Base**	**Train**	**Val**	**Test**
Person (wildcard)	401	280	80	41
Person (pose+wildcard)	126	88	25	13
Wild boar (wildcard)	567	397	113	57
Duck (wildcard)	181	126	36	19
Turtle (wildcard)	155	108	31	16
Background (empty)	74	52	14	8
Total	**1504**	**1051**	**299**	**154**

Table 2: YOLO11n detection performance on the synthetic validation set (299 images).

Class	**Precision**	**Recall**	**mAP@0.5**	**mAP@0.5:0.95**
Boar	0.930	0.925	0.958	0.734
Duck	0.913	0.760	0.867	0.401
Person	0.898	0.888	0.932	0.657
Turtle	0.908	0.781	0.892	0.411
All	**0.912**	**0.839**	**0.912**	**0.551**

filtered with NMS (confidence 0.7, IoU 0.5). At 12.8 ms/frame the Jetson comfortably serves both cameras at the required one frame per second, powered by a 500 W solar panel and a 100 Ah $LiFePO_4$ battery.

4 Cloning the Rescue Craft's Radio Link

The actuation half turns the consumer rescue buoy into a computer-commandable device *without touching its hardware*. We first observe what its remote transmits, then reproduce it.

4.1 The craft, and sniffing its SPI bus

The rescue buoy (Fig. 1) is a ≈105 cm twin-hull flotation body with two thrusters and a grab handle; a person in the water holds the handle and is kept afloat or towed. It is controlled by a hand-held remote whose core is an **STM32** microcontroller driving an **RFM98** module—a Semtech **SX1278** sub-GHz transceiver [15]—over a 4-wire **SPI** bus, with seven controls: *power*, *unlock*, the four directions and *lock*. Since the receiver expects this specific radio, the cleanest approach is to reproduce the remote's air interface verbatim.

Rather than demodulate the 433 MHz signal, we probe the digital bus *before* modulation, where the air payload is still in clear. We wired a logic analyser

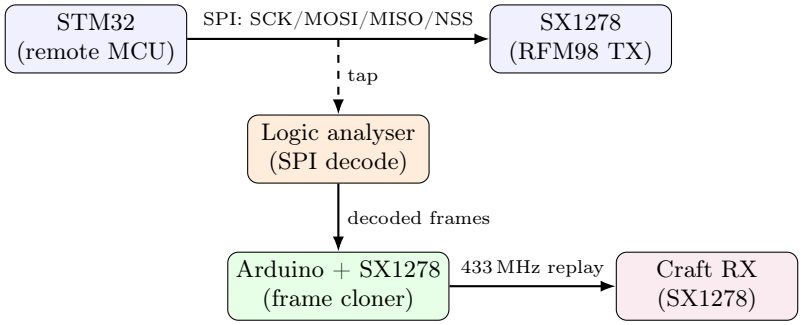


Fig. 3: Bus-level reverse engineering. The logic analyser taps the SPI link that feeds the transmitter, so the air payload is captured in clear; a second microcontroller then rebuilds and re-emits the same frames.

Table 3: Decoded 11-byte beacon frame. Bytes 0–1 are a fixed sync word excluded from the checksum; byte 10 is XOR of bytes 2–9.

Offset	**Bytes**	**Field / meaning**
0–1	`48 56`	Sync word ("HV"), fixed, outside the checksum
2	`06`	Payload length (6 bytes: 2 channels + 2 flags)
3	`A1`	Transmitter (remote) identifier
4–5	`CH1` (BE16)	Vertical axis, 1000–1500–2000 μs
6–7	`CH2` (BE16)	Horizontal axis, 1000–1500–2000 μs
8	`b8`	Heartbeat bit (toggles `0x80`/`0x00`)
9	`b9`	Lock latch (`0x01` unlocked / `0x00` locked)
10	`crc`	Checksum, XOR of bytes 2–9

to the four SPI lines (SCK, MOSI, MISO, NSS) between the STM32 and the SX1278 and recorded a session pressing every button in a known order (Fig. 3). On the SX1278, every over-the-air payload is a *burst write to the FIFO* (address `0x80`). Of the 1,809 SPI transactions in the capture, 132 are radio frames; the rest are start-up configuration and status polling.

4.2 Decoding the frame and the commands

Every radio frame is a fixed **11-byte** structure emitted as a continuous *beacon* every $\approx$179 ms (5.6 Hz), whether or not a button is held. Brute-forcing the last byte revealed it to be the XOR of bytes 2–9, which fixes the field layout (Table 3). The two 16-bit big-endian channels use the classic RC pulse range 1000–1500–2000 μs (`0x03E8`–`0x05DC`–`0x07D0`), centred at `0x05DC`. Byte 8 toggles between `0x80` and `0x00` in blocks of a few frames independently of any button: it is a link-alive heartbeat, not a control. Byte 9 is a lock latch.

Segmenting the trace by the known button order yields an unambiguous mapping (Table 4). *Power-on* has no air payload: it is the SX1278 boot sequence

Table 4: Joystick-to-frame mapping recovered from the capture.

Action	**Frame signature**
Power-on	SX1278 boot sequence (LoRa → standby → TX)
Unlock	byte 9: `00` → `01`
Up / Down	`CH1` > / < centre (`07D0` / `03EF`)
Left / Right	`CH2` > / < centre (`07C8` / `03E8`)
Lock	byte 9: `01` → `00`

(LoRa mode, carrier, PA config, TX) after which the beacon starts. *Unlock* flips byte 9 from `00` to `01`, and the four directions drive `CH1` or `CH2` above or below centre; the directional frames appear *only* while the latch is `01`. *Lock* flips byte 9 back to `00`.

4.3 Rebuilding the link and closing the loop

Reproducing a button therefore reduces to cloning an idle beacon, editing the relevant field and recomputing the XOR. We implemented this on an Arduino-class board with its own SX1278 module, wired over SPI with level shifting (the RFM98 is a 3.3 V part). At start-up the cloner replays the boot sequence: LoRa mode, the original carrier (`Frf` = `0x6DA000`), SF9/BW125/CR4-5 with CRC and payload length 11. Thereafter a `buildPacket()` routine assembles the 11 bytes—sync `48 56`, id `A1`, the two channels, heartbeat and latch, then XOR over bytes 2–9—and each command emits a burst of frames at the native ≈179 ms cadence. The abstract vocabulary of Sect. 2 maps one-to-one onto these calls; the firmware is released in the `mando_barco` repository [5].

The approach policy consumes the detector's target position and issues `LEFT`/`RIGHT` pulses to align the craft's heading with the victim and `UP` pulses to advance, re-evaluating on each detection. Two safety provisions follow from the protocol: the autonomous path emits `LOCK` whenever no target is tracked (craft idles instead of drifting), and a simple priority rule lets the hand-held remote pre-empt autonomous commands at any time.

5 Experimental Results

We evaluate the two halves separately: the detector in a real reservoir, and the cloned link on the bench and on open water.

5.1 Detection in the field

The perception node was deployed on a ≈1,500 m^2 reservoir in southern Spain. People were detected by introducing volunteers under varied conditions (day/night, clear/overcast, centre/edge/partially submerged); ducks and turtles were validated in a larger natural pond; and, since wild boars cannot be made to fall

Table 5: Field detection results (single-frame). Persons were tested in the target reservoir; boars on cross-environment footage.

Scenario	**Trials**	**TP**	**FP**	**FN**	**Rate (%)**
Person	190	178	6	20	93.6
daytime	140	134	3	12	95.7
nighttime	50	44	3	8	88.0
Wild boar	100	84	6	16	84.0
Overall	**290**	**262**	**12**	**36**	**90.3**

Table 6: Cloned-link validation. Field metrics (range, latency, time-to-reach) are preliminary and must be confirmed.

Metric	**Value**
Frame structure reproduced	11/11 bytes
Controls reproduced	7/7 (power, unlock, 4 dir., lock)
Beacon cadence	≈179 ms (5.6 Hz)
Carrier (from `Frf`)	≈438.5 MHz
CRC acceptance at receiver	100% (0/5000 rejected)
Control range (line of sight, water)	≈30 m
Time to reach a target buoy	<2 s (reservoir)

on demand, boar detection was scored on real footage from other water bodies. Over 290 single-frame trials the overall detection rate was 90.3% (Table 5): persons reached 93.6% (95.7% by day, 88.0% by night), while the cross-environment boar test was hardest at 84.0%.

5.2 Fidelity and responsiveness of the cloned link

Bench validation compared the cloner output byte-by-byte against captured originals (Table 6). Across 5,000 frames all fields matched in 100% of cases (0 CRC rejections), all seven controls were reproduced, and command-to-motion latency was dominated by the 179 ms beacon period.

While preliminary, this shows the perception node and the reverse-engineered radio together close the detect-and-recover loop.

6 Conclusions and Future Work

We have presented an end-to-end system that does not merely *detect* falls into irrigation reservoirs but begins to *respond* to them. Its perception half runs a YOLO11 detector trained on a reservoir-anchored synthetic dataset and reaches 0.91 mAP@0.5 and 90% field detection rate. Its actuation half reverse-engineers the buoy's proprietary RF link via SPI sniffing and rebuilds it on a commodity

microcontroller with no hardware change. Coupling the two closes the detect-and-recover loop while leaving a human able to take over at any time.

Several directions remain. On perception, a geographically diverse background set and more species would broaden cross-environment coverage. On actuation, closed-loop guidance via camera or GPS on the craft and systematic validation across weather conditions are the main open items. Finally, the bus-sniffing methodology generalises to other SX127x-based devices, suggesting a reusable recipe for repurposing closed RC hardware as robotic actuators.

Acknowledgments. This work has been supported by grants CPP2021-008931 and PID2022-137344OB-C32, funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR and "ERDF A way of making Europe". The project is carried out in collaboration with the company ATARFIL SL.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alotaibi, A.: Automated and intelligent system for monitoring swimming pool safety based on the IoT and transfer learning. Electronics **9**(12), 2082 (2020). https://doi.org/10.3390/electronics9122082
2. Amer, D.A., Ibrahim, N.Y., Ibrahim, I.K., Mohamed, A.M., Soliman, S.A.: Intelligent eyes on water: YOLOv11-based real-time drowning detection system. The Journal of Supercomputing **81**, 1242 (2025). https://doi.org/10.1007/s11227-025-07732-7
3. Bai, B., Yue, H., Chen, L., Li, X.: Research on dataset generation and monitoring of generative AI for drowning warning system. IEEE Access **12**, 83589–83599 (2024). https://doi.org/10.1109/ACCESS.2024.3407245
4. Black Forest Labs: FLUX. `https://github.com/black-forest-labs/flux` (2024)
5. Grupo AVISPA: mando_barco: Spi-level reconstruction of an RFM98/SX1278 rc link for a rescue craft. `https://github.com/grupo-avispa/mando_barco` (2026)
6. Jegham, N., Koh, C.Y., Abdelatti, M., Hendawi, A.: Evaluating the evolution of YOLO models: A comprehensive benchmark study of YOLO11 and its predecessors (2024), arXiv:2411.00201
7. Jocher, G., Qiu, J.: Ultralytics YOLO11. `https://github.com/ultralytics/ultralytics` (2024)
8. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., et al.: Segment anything. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3992–4003 (2023). https://doi.org/10.1109/ICCV51070.2023.00371
9. Lipman, Y., Chen, R.T.Q., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling (2023), arXiv:2210.02747
10. Liu, Z., Zhang, Y., Yu, X., Yuan, C.: Unmanned surface vehicles: An overview of developments and challenges. Annual Reviews in Control **41**, 71–93 (2016). https://doi.org/10.1016/j.arcontrol.2016.04.018

11. López-Felices, B., Aznar-Sánchez, J.A., Velasco-Muñoz, J.F., Piquer-Rodríguez, M.: Contribution of irrigation ponds to the sustainability of agriculture. a review of worldwide research. Sustainability **12**(13), 5425 (2020). https://doi.org/10.3390/su12135425
12. Martín Cazorla, F., Sánchez Blanque, J.L., Santos Amaya, I.M.: Muertes en balsas de riego en la provincia de almería. Revista Española de Medicina Legal **37**(4), 134–139 (2011). https://doi.org/10.1016/S0377-4732(11)70079-7
13. Mattoussi, W., Mattoussi, F., Zeddini, Y.: Does dam-based irrigation affect the sustainability of natural capital?: A doubly robust analysis. Journal of Cleaner Production **450**, 141764 (2024). https://doi.org/10.1016/j.jclepro.2024.141764
14. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models (2021), arXiv:2112.10752
15. Semtech Corporation: SX1276/77/78/79 – 137 MHz to 1020 MHz low power long range transceiver. Tech. Rep. Rev. 7, DS_SX1276-7-8-9_W_APP, Semtech Corporation (2020)
16. Shatnawi, M., Albreiki, F., Alkhoori, A., Alhebshi, M.: Deep learning and vision-based early drowning detection. Information **14**(1), 52 (2023). https://doi.org/10.3390/info14010052
17. Tudela, A., Ruiz-Beltrán, C.A., Pons, Ó., González-García, M., Bandera, A.: Wildlife in irrigation ponds dataset. `https://huggingface.co/datasets/grupo-avispa/wildlife_in_irrigation_ponds` (2025)
18. Tudela, A., Ruiz-Beltrán, C.A., Pons, O., González-García, M., Bandera, A.: Edge-computing for the early detection of falls by people and/or animals in reservoirs. Applied Sciences **16**(12) (2026). https://doi.org/10.3390/app16126117
19. Xiao, B., Wu, H., Xu, W., Dai, X., Hu, H., Lu, Y., Zeng, M., Liu, C., Yuan, L.: Florence-2: Advancing a unified representation for a variety of vision tasks. arXiv preprint arXiv:2311.06242 (2023)
20. Yang, Z., Zeng, A., Yuan, C., Li, Y.: Effective whole-body pose estimation with two-stages distillation. In: 2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). pp. 4212–4222 (2023). https://doi.org/10.1109/ICCVW60793.2023.00455
21. Z-Image Team: Z-image: An efficient image generation foundation model with single-stream diffusion transformer. arXiv preprint arXiv:2511.22699 (2025)

Real-time Cosmetic Contact Lens Detection System for Iris Recognition at a Distance

Camilo A. Ruiz-Beltrán[1 [0000−0001−8270−2062]], Adrián Romero-Garcés[1 [0000−0002−6570−5859]], Alberto Tudela[1 [0000−0001−6796−5286]], José Galeas[1 [0009−0007−9978−8406]], and Antonio Bandera[1 [0000−0003−3147−0307]]

Dpto. de Tecnología Electrónica, E.T.S. de Ingeniería de Telecomunicación, Universidad de Málaga, Málaga, Spain
{camilo,argarces,ajtudela,jgaleas,ajbandera}@uma.es

Abstract. Iris recognition is currently regarded as one of the most promising biometric method and has been applied in numerous fields. However, its deployment in certain real-world environments faces a number of challenges. Among these, it is worth noting that most solutions rely on software running on a dedicated computer, which is considerably large, expensive and energy-intensive. Furthermore, to be truly reliable, they must address the issue of presentation attacks, confirming that the segmented irises are genuine. This article presents a hardware-based solution that integrates real-time iris segmentation and contact lens detection. To achieve this, the system comprises three stages. The first stage uses a YOLOX network trained to detect two classes: eyes and iris/pupil. These two classes overlap in the latter, a fact which is exploited to prioritise the detection of the 'iris/pupil' class. The second stage receives the eye subimages provided by the first stage and employs a lightweight U-Net network to segment the iris. Finally, the third stage detects the presence of cosmetic contact lenses using a support vector machine (SVM) classifier based on a single BSIF (Binarised Statistical Image Feature) feature. Designed to operate in an 'Iris At A Distance' (IAAD) scenario, the whole system has been integrated into a single Multiprocessor System-on-Chip (MPSoC), utilising AMD's Deep Learning Processing Unit (DPU). The solution is capable of processing the more than 47 frames per second provided by a 16 Mpx CMOS digital image sensor, delivering satisfactory results in terms of iris segmentation and the detection of cosmetic contact lenses.

Keywords: Iris segmentation · Presentation attack detection · Cosmetic contact lens detection · Iris At A Distance.

1 Introduction

Biometric identification using iris recognition is based on utilising the pattern of the coloured, textured part of the eye (the iris) as the person's 'signature'. Among its main advantages, the uniqueness and stability of the iris pattern over time are particularly noteworthy [2]. Furthermore, it is a biometric pattern that

can be obtained using computer vision, i.e. a non-invasive technique. However, as the process requires the iris pattern to meet high quality standards (in terms of contrast and/or sharpness), typical devices require that the user position their eye at a short and precise distance from the sensor. Despite the power of this identification technique, it can be circumvented by using simple textured contact lenses. When placed over the iris, these lenses mask the original pattern and allow the system to be deceived. The detection of presentation attacks (PAD) is normally based on analysing the texture of the iris, training the system to distinguish between the texture associated with a normal iris and the one modified by the presence of the contact lens.

If this technique is to be used in a more flexible way (e.g. integrated into an IoT environment or on a mobile robot), user interaction should be minimised, allowing the camera to operate at a greater distance from the user, so that she does not have to remain completely still. The Iris At A Distance (IAAD) recognition postulates that this iris image will be captured from a person in motion and at an increasing distance (more than 1 meter) from the camera. Moreover, it would be useful to equip the system with the ability to detect the presence of textured contact lenses. Finally, in these scenarios, energy consumption may be an important requirement. This involves moving from a GPU-centred design to one based on lower-power devices, such as MPSoCs.

The proposal described in this article builds on previous work, in which the detection and segmentation system [8] and the system for detecting textured contact lenses [7] were designed. In fact, it can be regarded as an extension of the first of the cited studies, to which the PAD system has been added. This PAD system differs from the original proposal [7] in that it reduces the number of SVM classifiers used from three to just one. The implications of this change will be described when the results are presented. The main contribution of this article is, therefore, the integration, within an MPSoC, of a complete framework for the segmentation and detection of fraud involving the use of cosmetic contact lenses. As will be demonstrated when the results are presented, the system is capable of processing the 47 frames per second provided by a 16 Mpx image sensor, with a power consumption—relative to the processing—that does not exceed 5.5 W.

The remainder of the paper is organised as follows: Section 2 provides an overview of the proposed methodology, detailing the acquisition setup and data collection. Section 3 describes the embedding of the whole framework in the MPSoC. Section 4 evaluates the accuracy of the proposed approach in terms of iris segmentation using publicly available databases. The resource utilisation and power consumption are also detailed. Section 5 concludes the paper and describes future work.

2 Methodology

The aim of the research is to integrate, within a single MPSoC, a complete system for iris detection and segmentation, and for detecting the presence of cosmetic

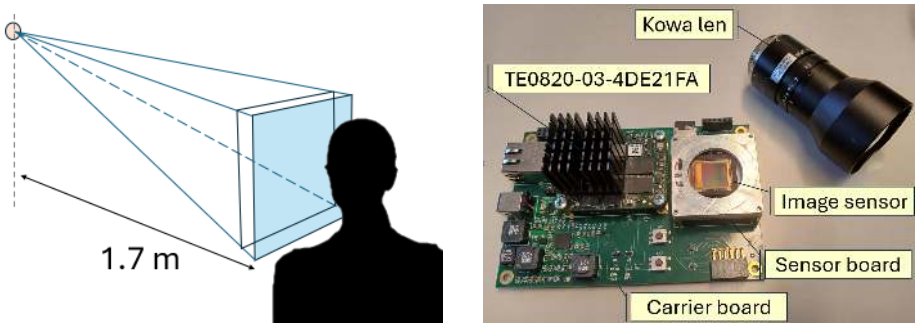


Fig. 1. (Left) The data acquisition setup; and (Right) the electronics of our system.

contact lenses. This is a significant challenge because, on the one hand, high quality standards must be maintained at every stage of the process, and, on the other, our system must be able to process the 47 images per second captured by a 16 Mpx CMOS image sensor. To do this, the system must carry out three processes which have traditionally consumed significant computational resources: detecting the area(s) containing the iris(es) in the input image; segmenting them to obtain normalised iris patterns; and analysing these to determine whether or not they belong to a cosmetic contact lens. The proposed architecture for addressing these issues recognises that the initial stages are particularly computationally intensive and, in order to process them in real time (47 fps), they have been designed to be accelerated using the programmable logic (PL) part of the MPSoC. Image capture from the sensor must also be implemented within the PL. The detection of contact lenses should be carried out on the normalised irises detected per frame. This final stage will run in the MPSoC's processing system (PS), using a single SVM classifier.

2.1 Data acquisition setup

The system has been designed to operate in an IAAD scenario: the image sensor is fixed and monitors a specific passageway through which people walk at a normal walking speed ($\approx$ 2 m/s). The detection zone is situated approximately 1.7 metres from the sensor, and the only requirement for the person is that she does not obstruct the capture of images of their eyes. The system uses near-infrared lighting (940 nm), which is triggered in sync with the image capture to avoid a hazardous exposure. Given the restrictions imposed by the system when capturing high-contrast images (short exposure time (1 ms) and aperture), the depth of field remains very shallow. All the 47 fps provided by the sensor must be processed in order to obtain around 5–8 in-focus irises for each person passing through the system. The data acquisition system is schematised in Figure 1 (Left).

Figure 1(Right) shows the electronics of our proposal. Briefly, we mounted on a custom-made carrier board a sensor board (interfacing the image sensor,

a Telecyne e2v EMERALD 16MP) and the Trenz TE0820-03-4DE21FA micromodule (porting the AMD/Xilinx Ultrascale+^{TM} ZU4EV MPSoC). The optics is a LM100JC1MS from Kowa. Lighting is provided by a high-power LEDs device triggered by the capture module.

2.2 Datasets

The three problems managed by our system require specific datasets for training. For the iris detection, a dataset of 7,701 images were captured and annotated. Two classes were labelled: eye and iris/pupil. Having both classes allows for robust detection of the latter: an iris/pupil region will only be valid if it is within an eye region. This dataset must include out-of-focus images, which will not be annotated as valid eyes or irises. At runtime, when a person's face is outside the range of distances defined by the depth of field, their eyes will appear out of focus and should not be detected. A second dataset is needed to train the iris segmentation stage. In this case, all inputs can be focused eyes. Thus, we used the CASIA-Iris-V3-Interval[1] and IIT Delhi 1.0 [6]. The ground truth for iris segmentation is available for both in the IRISSEG-CC (Halmstad University) and IRISSEG-EP (University of Salzburg) databases [1, 4], respectively. To these datasets we have added a set of 263 images captured by our system, whose ground truths have been manually generated. Finally, to train the PAD, we used the IIITD-CogentScanner dataset [10] and the NDCLD'15 [11].

All the material acquired with our system comes from the setup described above and was recorded from 15 volunteers walking through the access point at normal speed, without any restriction regarding glasses or make-up. Three disjoint subsets were derived from these sequences: (i) the 7,701 annotated frames used to train the detector, which deliberately include out-of-focus faces so that the network learns to reject them; (ii) 263 in-focus eye images, with manually annotated segmentation masks, added to the public training material of the segmentation stage; and (iii) 473 images from held-out sequences, reserved for the evaluation reported in Table 1. It should be noted that the captured sequences do not include subjects wearing cosmetic contact lenses. Consequently, and as discussed in Section 4, the PAD module has been trained and evaluated using only the public datasets.

3 The proposed approach

3.1 Iris detection and segmentation

Figure 2 provides an overview of the proposed architecture. The frames of the video stream provided by the EMERALD 16MP sensor are stored, in its original size, in the DDR memory, and resized to a size of 256 × 256 pixels for processing. The iris detection and segmentation tasks have been addressed using two inference models: a YOLOX model is employed to solve the iris detection, and

[1] CASIA-IrisV3, `http://www.cbsr.ia.ac.cn/IrisDatabase`

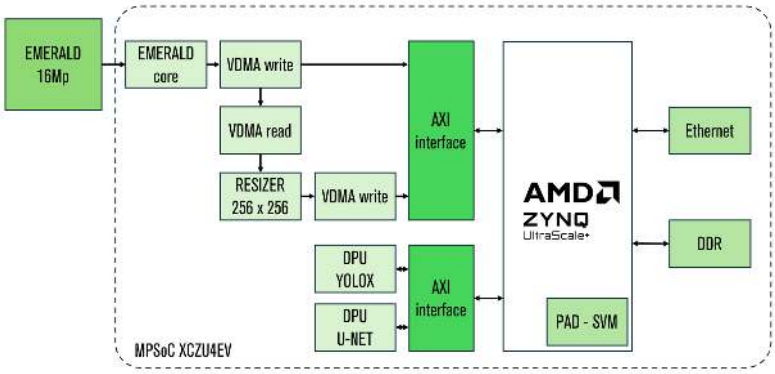


Fig. 2. Overview of the proposed architecture

a lightweight U-Net for solving the iris segmentation. Both models are implemented using DPU (Deep learning Processor Unit) IP cores from AMD/Xilinx [8]. This core is a configurable design-time accelerator that, synthesized in the PL part of the MPSoC, is involved from the PS part to process a convolutional neural network layer by layer. Thus, the first DPU takes as input the scaled image stream and accelerates the YOLOX model. It is able to process 47 fps, providing a stream of probable iris images. A contrast and sharpness test filters out images with low values, and the resulting stream is sent as input to the second DPU. This DPU accelerates a lightweight U-Net model. It is capable of processing 16 images per second, a figure considered sufficient as there will be few images containing correctly focused eyes after a person has passed by. Once the iris has been segmented, the subsequent steps — which consist of obtaining the normalised rectangular iris pattern; determining whether or not a textured contact lens is present; and sending the iris pattern and fraud information for final processing (recognition and/or alarm generation) — are carried out on the PS.

The detection is addressed using a modified version of the YOLOX nano [3]. In short, we reduce the input layer to accept 256×256 input images, and the activation function was modified from SiLU (Sigmoid Linear Unit) to the simpler ReLU (Rectified Linear Unit). The model was trained using QAT (Quantization-aware Training). As aforementioned, the model detects two classes: eyes and irises/pupils. A post-processing step is added to discard any eye detections whose bounding boxes do not contain a positive detection of the iris/pupil. Then, a second post-processing step filters out those images exhibiting a low value of sharpness and/or contrast. Only images that exceed a threshold of contrast and sharpness are provided to the iris segmenter. This detection phase is key to the system: the number of frames passing through to the subsequent stages must be reduced as much as possible, whilst also minimising the risk of incorrectly discarding an eye whose iris might enable the subject to be recognised. Table 1 shows the results obtained by this phase on a dataset with 2,392 images from the

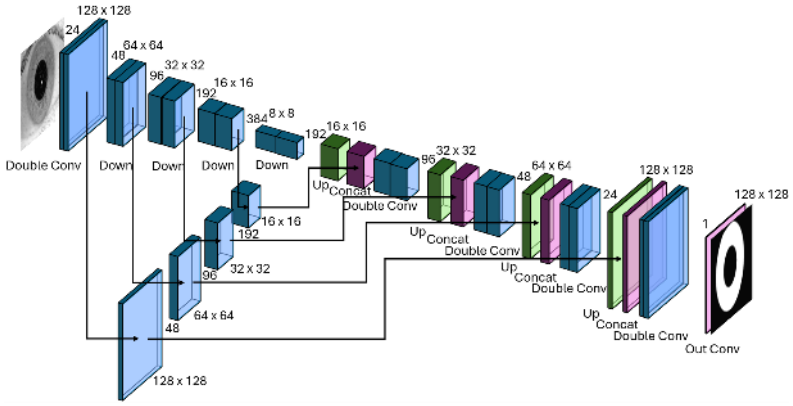


Fig. 3. The U-Net architecture

CASIA and IITD databases, and on a dataset of 473 images acquired with our system. In this second dataset there are images that are not valid for recognition (TN) because they are out-of-focus.

Table 1. Evaluation results of the proposed filter on different datasets.

Dataset	TP	TN	FP	FN	Precision / Recall / F1-score
CASIA/IITD	2392	0	0	48	1.000 / 0.980 / 0.990
OUR	46	427	6	1	0.885 / 0.979 / 0.929

The composition of the OUR set illustrates the role of this stage: the 473 images correspond to complete passes of people through the access point and are dominated by frames that must be rejected. Only 46 of them contain an iris that is valid for recognition (TP), whereas 427 are out-of-focus or otherwise unusable (TN). The filter therefore discards more than 90% of the incoming eye detections while retaining 46 of the 47 usable irises (a single FN), at the cost of 6 false positives that correspond to borderline-focus images. We are aware that the number of positive samples in this set is small, so these figures characterise the operating point of the filter rather than providing a definitive accuracy estimate.

Iris segmentation is accomplished using a modified, lightweight version of the U-Net architecture [9]. Thus, the proposed model achieves a computational cost of 1.93 GFLOPS per forward pass. Figure 3 illustrates the architecture of the modified U-Net. The model was quantized to 8-bit integers using symmetric power-of-two quantization.

Once the image pixels corresponding to the iris region has been marked, it is needed to encode in a normalised iris pattern. This pattern is a rectangular shaped-region, consisting of iris and non-iris pixels. The iris pixels are characterised by the brightness values of the original captured image. The non-iris

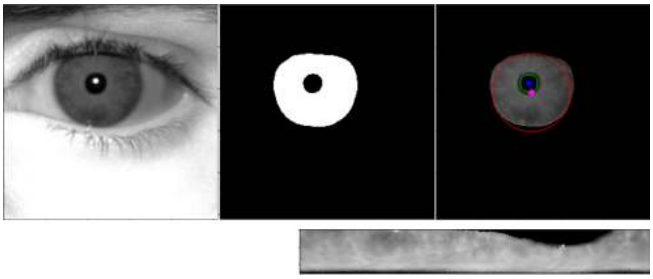


Fig. 4. (Top) After obtaining the segmentation mask associated to an eye image, we need to estimate the center and radius of the pupil and iris regions; (Bottom) the remapping of the iris region from Cartesian coordinates to a normalised polar representation provides the normalised iris pattern.

ones are encoded with a specific value. Figure 4 shows the process for obtaining the normalised iris pattern associated to a detected eye image. As a first step, we need to estimate the center and radius of the iris and pupil regions. Then, the segmented iris region undergoes Daugman's rubber sheet normalization [2], remapping the iris region from Cartesian coordinates to a normalized polar representation. This transformation provides invariance to pupil dilation and iris size variation, being essential for enabling robust and consistent feature extraction across different subjects and imaging conditions. All these post-segmentation processing steps run in the PS part of the MPSoC.

To satisfy real-time constraints, the stage involving the eye detection and contrast/sharpness filters runs concurrently with the one solving the preprocessing, iris segmentation, radius/centre estimation, and iris unroll. Both processes communicate via sockets. Although, in the second stage, a circular buffer with a capacity of 2000 samples was implemented, this was never observed to reach its maximum capacity in our experiments.

3.2 Presentation attack detection

The normalised iris pattern is the input image for conducting the iris PAD. The approach consists of three steps. Firstly, the iris pattern is encoded using one BSIF code. Then, the z-score normalised histogram associated to this BSIF code is obtaining. Finally, the presence or not of a cosmetic contact lens is determined using a SVM classifier.

The BSIF descriptor computes a binary code string $\mathbf{b}$ for all pixels of the input image. Each bit of the code string is calculated by binarising the output of a linear filter, with a threshold at zero. Thus, there are two parameters in a BSIF encoding: the length n of the code string (i.e. the number of filters), and the size $(s \times s)$ of the filter. For instance, a BSIF-12-18 is obtained using 12 filters of size 18×18. Given an image patch I of size $s \times s$ pixels and n linear filters

$\{W_k\}_{k=1...n}$ of the same size of I, the vector $\mathbf{x}$ of filter responses is obtained by

$$\mathbf{x} = (\sum_{i,j} W_1(i,j)I(i,j), \sum_{i,j} W_2(i,j)I(i,j), ... \sum_{i,j} W_n(i,j)I(i,j)) \quad (1)$$

The binary code string $\mathbf{b}$ is computed by setting b_i equal to 1 if the corresponding $x_i > 0$ and to 0 otherwise. Once the BSIF code histogram for an iris pattern is obtained, its bins are normalised to have a mean value equal to 0 and a standard deviation of 1 (z-score normalisation). In short, the code string provided by the BSIF is a local descriptor of the image intensity pattern in the vicinity of the pixel, and the histogram of code values is then a mechanism for encoding the texture properties of an image.

To determine which BSIF features and classifier to use as an iPAD, all possible combinations of BSIF features were constructed (with n ranging from 5 to 12, and s from 3 to 34) and, using the resulting histograms, two classifiers (SVM and Multi-Layer Perceptron) were evaluated NDCLD'15 as training dataset. The best solution (BSIF-12-18-svm) was used to be integrated in our proposal.

4 Experimental results

The system has been intensively tested in a real scenario, where natural lighting is not controlled and where people passing by the access point do not cooperate with the system. They are simply asked to walk past. The access point receives natural light, whose level and direction change with the time of day and the weather, and the sequences employed in this work were recorded in several sessions under such varying conditions. Robustness to this variability is mainly obtained from the acquisition design rather than from the inference models: the 940 nm illuminator is fired in sync with a short exposure time (1 ms), so the contribution of the ambient light to the captured image is small and the appearance of the iris texture remains stable. A systematic characterisation of the performance as a function of the ambient illuminance has not been conducted and is part of our future work.

Figure 5 shows some snapshots of a passing sequence. At 47 fps, if the person is in front of the image sensor during three seconds, the detection stage should provide a set of 200-250 eye images to the rest of the stages. This will provoke the collapse of the framework. However, in our case, the contrast/sharpness filters discard a significant number of images. Thus, the number of images that are passed to the segmentation stage typically does not exceed 10-15 detections. In the figure, the detection of eyes and irises/pupils only provides valid detections in those images that are in-focus. We can also see that there is always an iris/pupil detection (marked in yellow) within an eye detection (marked in blue). Averaged over the recorded passes, this chain (200–250 eye detections, 10–15 of them accepted by the contrast/sharpness filter) ends up providing 5–8 normalised iris patterns per person, which are the samples finally available for recognition and for the PAD.

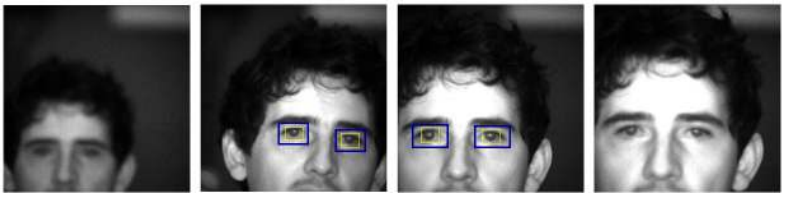


Fig. 5. Eye detection using the proposed approach

The stream of detected eye images is the input to the segmentation stage. Using the average DICE coefficient to measure the similarity between the ground-truth masks and the obtained iris segmentations, our proposal provides an average DICE coefficient of 0.9706. This value is higher than that of other more complex approaches, with a number of parameters that is also lower than that required by other methods [8].

Although an instrumented measurement of the end-to-end response time was not an objective of this work, it can be bounded from the throughput of the stages. The capture and detection stage delivers a decision every 21 ms (47 fps) and the segmentation DPU processes one eye image every 62.5 ms (16 fps). The post-processing running on the PS (centre and radius estimation, unrolling, BSIF encoding and SVM classification) does not constrain this figure, as the circular buffer feeding the second stage never reached its maximum capacity, i.e. no accumulation was observed at its input. Hence, the delay from the frame in which an eye is detected to the availability of its normalised pattern, labelled as genuine or attack, stays below 100 ms, so the result is produced while the person is still in front of the sensor.

Finally, the iPAD provides a correct classification rate of 96.5% when tested with the IIIT-Cogent Scanner dataset. However, as this was the database used for training, we have also evaluated the proposal using the complete IIITD database. In this case, the proposal provides a rate of 92.7%. This value is less than the one obtained when three SVMs are employed (in this case, the rate was of 94.65%)[7], but the model is clearly more lightweight. Moreover, we should consider that these results are derived from the processing of a single iris. As the framework is able to manage 5-6 in-focus irises per person, the probability that the contact lens will be detected in at least one of these n patterns would be

$$P(detection) = 1 - (1 - p)^n \tag{2}$$

p being the probability of obtaining a correct detection in one image. The equation assumes that the n classification processes are independent, as is the case in our system. With n equal to 3, and the 0.927 value for p, the value of the cumulative probability of correctly classifying the iris pattern (cosmetic contact lens or not) is 99.96%. The choice of $n = 3$ is deliberately conservative with respect to the 5–8 patterns typically obtained per pass.

It must be emphasised that these PAD rates were obtained on public datasets acquired with commercial close-up sensors, and not on data captured by the

complete system, as our sequences do not include subjects wearing cosmetic contact lenses. Since the normalised patterns obtained in an IAAD scenario are noisier than those of the public datasets, a degradation of the per-image rate should be expected in operation, which Equation (2) partially compensates. A dedicated acquisition campaign, with volunteers wearing cosmetic lenses passing through the access point, is the natural continuation of this work and the only way to validate this module end to end.

Table 2 summarises the comparison with the two works that this proposal extends and integrates. A broader quantitative comparison with other embedded FPGA/MPSoC solutions is not straightforward: published hardware implementations usually address a single stage of the pipeline (detection or segmentation), assume cooperative close-up acquisition, and do not report throughput, accuracy and power consumption on a common basis. To the best of our knowledge, no other embedded system integrates detection, segmentation and cosmetic contact lens detection for an IAAD scenario on a single device.

Table 2. Comparison with the previous works integrated in this proposal (n/r: not reported).

Approach	Platform	Stages	Throughput	PAD CCR
[7]	PC (software)	PAD (3 SVMs)	n/r	94.65%
[8]	ZU4EV MPSoC	Det. + segm.	45 fps	–
This work	ZU4EV MPSoC	Det. + segm. + PAD	47 fps	92.7% (99.96%*)

* cumulative value over $n = 3$ patterns, Equation (2). Power: 5–5.2 W (this work).

One of the most critical topics of this work is to optimise the hardware resource consumption. Synthesized in the XCZU4EV-1SFVC784I, Table 3 shows the resource utilisation for the proposed architecture. It can be seen how some of the resources are employed in a high percentage (LUT, URAM, FF). Significantly, the design is able to process the 47 fps captured from the sensor operating at 150 MHz (only the DSPs run at 300 MHz). This has allowed us to avoid an excessive heating of the whole device. The power consumption of the proposed design is maintained in the range 5 to 5.2 W.

Table 3. Resource utilisation for the proposed design

Resource	Available	Utilisation
LUT	87,840	80,813 (92%)
FF	175,680	126,490 (72%)
BRAM	256	72 (28%)
URAM	48	48 (100%)
DSP	728	459 (63%)
MMCM	4	1 (25%)

5 Conclusions and future work

This paper shows the design and implementation of a complete iris capture, detection and segmentation system, endowed with the ability of detecting the presence of cosmetic contact lens. The whole system is embedded in a single MPSoC, ensuring low power consumption and industrial-grade robustness. To deal with the detection and segmentation issues, two inference models were designed: a YOLOX for iris detection and a lightweight UNet for iris segmentation. The iPAD is implemented in the PS of the MPSoC as it run using a single SVM classifier. The system has successfully shown its capacity to manage the IAAD scenario, where normalised iris patterns are extracted from moving people passing through an access point without cooperating with the system. These patterns are demonstrated to be valid for iris recognition.

The main limitations of the present evaluation are the reduced size of the set captured with our system (473 images, 46 of them valid for recognition) and the fact that the PAD module has been assessed on public datasets rather than on patterns extracted by the complete framework. Enlarging the captured dataset, including volunteers wearing cosmetic contact lenses, and instrumenting the end-to-end latency and the behaviour under controlled changes of the ambient illuminance, are therefore the priorities of our current work.

Further extension of this work should consider to include the iris recognition task within the MPSoC. One option is to consider a solution that integrates iris segmentation, feature extraction and matching into a unified model [5]. The problem with these solutions is that they are more difficult to debug, as many tasks are mixed in a single network. Therefore, we are studying the option of moving the design to a platform with more resources in the PL, and including recognition as a third inference model (accelerated using a third DPU).

Acknowledgments. This work has been supported by grants CPP2021-008931 and PID2022-137344OB-C32, funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR (for the first one), and "ERDF A way of making Europe" (for the second one). Portions of the research in this paper use the CASIA-IrisV3 and CASIA-IrisV4 collected by the Chinese Academy of Sciences—Institute of Automation (CASIA). A dataset was also captured with the proposed system in a real scenario. We thank Yael Casquet, Ana Barrio, and Paula M. Lozano, for their work annotating the captured sequences.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alonso-Fernandez, F., Bigun, J.: Near-infrared and visible-light periocular recognition with gabor features using frequency-adaptive automatic eye detection. IET Biometrics **4**, 74–89 (2015). https://doi.org/10.1049/iet-bmt.2014.0038

2. Daugman, J.: How iris recognition works. IEEE Transactions on Circuits and Systems for Video Technology **14**(1), 21–30 (2004). https://doi.org/10.1109/TCSVT.2003.818350
3. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021 (2021), https://arxiv.org/abs/2107.08430
4. Hofbauer, H., Alonso-Fernandez, F., Wild, P., Bigun, J., Uhl, A.: A ground truth for iris segmentation. In: 2014 22nd International Conference on Pattern Recognition. pp. 527–532 (2014). https://doi.org/10.1109/ICPR.2014.101
5. Hu, Q., Yin, S., Ni, H., Huang, Y.: An end to end deep neural network for iris recognition. Procedia Computer Science **174**, 505–517 (2020). https://doi.org/https://doi.org/10.1016/j.procs.2020.06.118, https://www.sciencedirect.com/science/article/pii/S1877050920316380, 2019 International Conference on Identification, Information and Knowledge in the Internet of Things
6. Kumar, A., Passi, A.: Comparison and combination of iris matchers for reliable personal authentication. Pattern Recognition **43**(3), 1016–1026 (2010). https://doi.org/10.1016/j.patcog.2009.08.016
7. Romero-Garcés, A., Ruiz-Beltrán, C., Marfil, R., Bandera, A.: Lightweight cosmetic contact lens detection system for iris recognition at a distance. In: García Bringas, P., Pérez García, H., Martínez de Pisón, F.J., Martínez Álvarez, F., Troncoso Lora, A., Herrero, Á., Calvo Rolle, J.L., Quintián, H., Corchado, E. (eds.) 18th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2023). pp. 246–255. Springer Nature Switzerland, Cham (2023)
8. Ruiz-Beltrán, C., Pons, O., González-García, M., Bandera, A.: Real-time detection and segmentation of the iris at a distance scenarios embedded in ultrascale mpsoc. Electronics **14**(18) (2025). https://doi.org/10.3390/electronics14183698
9. Safarov, F., Khojamuratova, U., Komoliddin, M., Kurbanov, Z., Tamara, A., Nizamjon, I., Muksimova, S., Cho, Y.I.: Lightweight evolving u-net for next-generation biomedical imaging. Diagnostics **15**(9) (2025). https://doi.org/10.3390/diagnostics15091120
10. Yadav, D., Kohli, N., Doyle, J.S., Singh, R., Vatsa, M., Bowyer, K.W.: Unraveling the effect of textured contact lenses on iris recognition. IEEE Transactions on Information Forensics and Security **9**(5), 851–862 (2014). https://doi.org/10.1109/TIFS.2014.2313025
11. Yambay, D., Becker, B., Kohli, N., Yadav, D., Czajka, A., Bowyer, K.W., Schuckers, S., Singh, R., Vatsa, M., Noore, A., Gragnaniello, D., Sansone, C., Verdoliva, L., He, L., Ru, Y., Li, H., Liu, N., Sun, Z., Tan, T.: Livdet iris 2017 — iris liveness detection competition 2017. In: 2017 IEEE International Joint Conference on Biometrics (IJCB). pp. 733–741 (2017). https://doi.org/10.1109/BTAS.2017.8272763

From Prototype to Classroom: Robotito, an Open-Source ROS 2 Teaching Robot

Guillermo J. Martos Aguilera[1 [0009−0002−0070−5759]], José Galeas[1 [0009−0007−9978−8406]], Óscar Pons[1 [0009−0002−6518−0666]], Alberto Tudela[1 [0000−0001−6796−5286]], and Juan P. Bandera Rubio[1 [0000−0003−3814−0335]]

Dpto. de Tecnología Electrónica, E.T.S. de Ingeniería de Telecomunicación, Universidad de Málaga, Málaga, Spain
guillermo.martos,jgaleas,opfernandez,ajtudela,jpbandera@uma.es

Abstract. Commercial ROS 2 differential robots are excellent teaching tools, but their cost limits laboratory fleet sizes and per-student hands-on time. We present *Robotito*, an open-source robot built around an ESP32 microcontroller and micro-ROS that runs full 2D SLAM and autonomous Nav2 navigation for under **200 €**. This yields a tenfold cost reduction compared to standard platforms like TurtleBot, while resolving critical safety and reliability defects (thermal runaway, battery instability, corrupted scans) of our baseline prototype, *Minibot v0*. Robotito couples PID-controlled geared DC motors, a protected Li-Ion pack and a 2D LiDAR, coordinated by a dual-core FreeRTOS/micro-ROS firmware that exposes the ESP32 as a native ROS 2 node. Validated in a domotic test apartment against twelve functional and non-functional requirements, it achieves sub-2 cm straight-line accuracy, a maximum surface temperature of 40.1 °C, and a bill of materials as low as 77 €. Hardware (KiCad, Fusion 360), firmware and navigation configuration will be released open-source for a new *Microbotics* course at the University of Málaga. Future work will address controlled benchmarks against alternative platforms, communication metrics, and classroom outcome data.

Keywords: Educational robotics · Differential drive · ROS 2 · micro-ROS · ESP32 · SLAM · Nav2 · Low-cost robot.

1 Introduction

Autonomous mobile robotics has moved in three decades from costly laboratory systems – e.g., the Pioneer research base (1996) – to affordable, increasingly capable commercial platforms, illustrated by the iRobot Roomba (2002), the TurtleBot family (from 2010) and, more recently, service robots such as PAL Robotics' TIAGo [12, 13]. Two forces drive this democratization: the mass production of MEMS sensors and 32-bit microcontrollers – the dual-core, WiFi-enabled ESP32 being a paradigmatic example [9] – and the maturation of ROS, from a centralized framework [11] into ROS 2, whose embedded extension micro-ROS ports a lightweight DDS implementation (Micro XRCE-DDS) down to the microcontroller [7, 3].

Despite this trend, the differential-drive platforms commonly used to teach ROS 2 remain expensive. The TurtleBot 3 Burger, the long-standing reference for makers and ROS education, costs around 500–700 €; the recent TurtleBot 4 costs over 1,195 USD and relies on an on-board Raspberry Pi 4 for ROS 2 computation [2]. Professional research bases such as the Pioneer 3-DX exceed a thousand euros, and service platforms like the TIAGo Base cost well over fifteen thousand [1]. Per-unit price therefore caps the number of robots per laboratory and, with it, practice time per student; a platform replicable for under 200 € would make one-robot-per-2–3-students labs, robot swarms and cooperative-robotics exercises economically feasible.

This work targets that gap with an engineering and educational focus, claiming no methodological novelty. Algorithms (SLAM Toolbox, AMCL, Nav2's Regulated Pure Pursuit) are used unmodified [5, 4, 6]. We provide a reproducible re-engineering of a non-functional prototype into a safe, low-cost platform running on a microcontroller, validated for unsupervised teaching. Quantitative benchmarking and classroom evaluations remain as future work.

Minibot v0, the ESP32-based differential prototype that motivated this work, proved the concept but was unusable as a laboratory tool (Fig. 1): its stepper motors overheated and deformed the printed chassis, its LiPo battery was unsafe for unsupervised student handling, and its LiDAR firmware produced mirrored, randomly rotated scans that made SLAM impossible. We re-engineer the prototype into *Robotito* (Fig. 2), a complete, safe and reproducible robot, and contribute:

- A **full hardware redesign** resolving the thermal, electrical and mechanical limitations of the baseline: geared DC motors with Hall encoders and H-bridge drivers instead of steppers, a protected 3S Li-Ion pack with a dedicated power-distribution stage, an inertial unit, and a custom, two-iteration modular PCB (perfboard to fabricated KiCad/JLCPCB board) with a documented expansion header (Sect. 4).
- A **real-time embedded firmware** that turns the ESP32 into a native ROS 2 node through micro-ROS, with class-based FreeRTOS task partitioning across the two cores, closed-loop PID odometry, a clock synchronization scheme and automatic reconnection (Sect. 5).
- A **fix of the LiDAR driver** and a **navigation stack** that couples SLAM Toolbox and Nav2 unmodified, plus an audiovisual human–robot interaction (HRI) module driven by a twelve-state, priority-based state machine (Sect. 6).
- An **experimental validation** of twelve functional and non-functional requirements in a realistic domotic environment, an explicit positioning of the platform against existing educational robots (Sect. 2), and an open release of hardware, firmware and CAD for a new university course (Sect. 7).

2 Related Work

Low-cost educational robots. Affordable differential platforms for teaching have a long tradition, from the Khepera and e-puck miniature robots to Arduino/Raspberry-

Table 1. Qualitative positioning of Robotito relative to TurtleBot 3/4. Prices are launch/list prices from the manufacturer/distributors; Robotito's is the authors' bill of materials (Table 4). Figures below the dashed line are the dimension this paper substantiates experimentally; figures above it are vendor specifications, included for context only.

	Robotito	**TurtleBot 3 Burger**	**TurtleBot 4 (Lite/Standard)**
On-board compute	ESP32 (dual-core MCU)	Raspberry Pi (SBC)	Raspberry Pi 4 (SBC)
Heavy computation	Off-board (laptop)	On-board	On-board
LiDAR	2D (Delta-2A)	2D	2D
Depth/RGB camera	—	optional	OAK-D-Lite/Pro
IMU / encoders	Yes / Yes (Hall)	Yes / Yes	Yes / Yes
Approx. unit cost	**77–145 €**	500–700 €	1,195–1,850 USD
Typical lab ratio	1 robot / 2–3 students	1 robot / group of groups	1 robot / group of groups

Pi rovers and the ubiquitous TurtleBot family [10, 2]. TurtleBot 3 set the template followed by most ROS-based teaching robots: a modular, 3D-printable, open-source differential base. Robotito follows the same philosophy but reduces cost by an order of magnitude and moves real-time control onto a microcontroller rather than an on-board SBC, exposing students to embedded development.

Positioning against TurtleBot 3/4. Table 1 contrasts Robotito and TurtleBot. While TurtleBot provides superior computation and sensing (SBC, depth camera), Robotito drastically reduces acquisition cost. Robotito complements standard platforms by facilitating affordable instruction in ROS 2, SLAM, and embedded programming. TurtleBot remains preferable for advanced perception or production-grade reliability. **A controlled, head-to-head benchmark** between both platforms is pending for future work (Sect. 8).

Microcontroller-class ROS nodes. Bridging microcontrollers to ROS was historically done with restrictive serial bridges such as `rosserial`. micro-ROS instead exposes an ESP32 as a first-class node in the ROS 2 graph over WiFi or serial [3], an approach adopted by several recent teaching and research robots [9]. Robotito contributes a fully documented, class-based firmware (Sect. 5) that runs motor control, LiDAR parsing and the micro-ROS executor concurrently under FreeRTOS, with explicit core pinning and clock synchronization that proved essential for a stable TF tree. While this partitioning employs standard producer/consumer patterns, it documents practical solutions to specific embedded timing pitfalls (clock drift, UART bursts) absent from typical micro-ROS tutorials.

SLAM and navigation in ROS 2. 2D LiDAR SLAM and Nav2 are mature in ROS 2: SLAM Toolbox provides pose-graph SLAM with loop closure [5]; Nav2 supplies a behaviour-tree-based stack with global planners, local controllers, and AMCL localization [8, 6, 4]. Robotito runs these algorithms unmodified on a genuine microcontroller-driven platform, including the firmware-level sensor corrections this required.

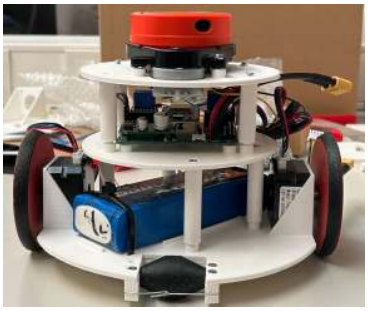


Fig. 1. *Minibot v0*, the non-functional baseline prototype.

Fig. 2. The re-engineered *Robotito* (final PCB revision), with managed wiring, modular PCB and the audiovisual HRI module.

3 Baseline Platform and Limitations

Minibot v0 (Fig. 1) is a 3D-printed, three-layer differential prototype following the TurtleBot philosophy, built around an ESP32-WROOM-32 module, a 3irobotix Delta-2A 2D LiDAR, two NEMA-17 stepper motors with TMC2208 drivers, and a 5000 mAh 3S LiPo battery, on a PCB with a 12 V→5 V buck stage. It demonstrated that an ESP32 can drive a differential robot, but four defects made it unusable for unsupervised teaching:

1. **Thermal runaway of the steppers.** Stepper motors hold phase current even at rest; combined with driver losses, this produced current peaks of up to 3 A at low speed and enough heat to *deform the PLA chassis* around the motor mounts, making the robot unsafe to handle.
2. **Unsafe battery.** The LiPo pack is chemically unstable under misuse, swelled with use (stressing the chassis) and represented about 30 % of the robot weight – unacceptable for frequent, unsupervised student handling.
3. **Broken LiDAR driver.** The Delta-2A driver published mirrored scans (points reflected about the longitudinal axis) and used an *inconsistent start angle* at every boot, yielding a randomly rotated map; both defects made SLAM impossible.
4. **Mechanical and integration issues.** Caster wheels rubbing on their housing, a loose battery mount, and unmanaged wiring.

These limitations define the redesign requirements (Table 3): thermal and electrical safety, cost under 200 €, consumer-printer reproducibility, deterministic WiFi control, modular expansion, and full SLAM navigation. The result, *Robotito* (Fig. 2), was developed through two iterations: a perfboard prototype (V1) for validation, and a custom JLCPCB two-layer board (V2). This process itself serves as a hardware prototyping example for students.

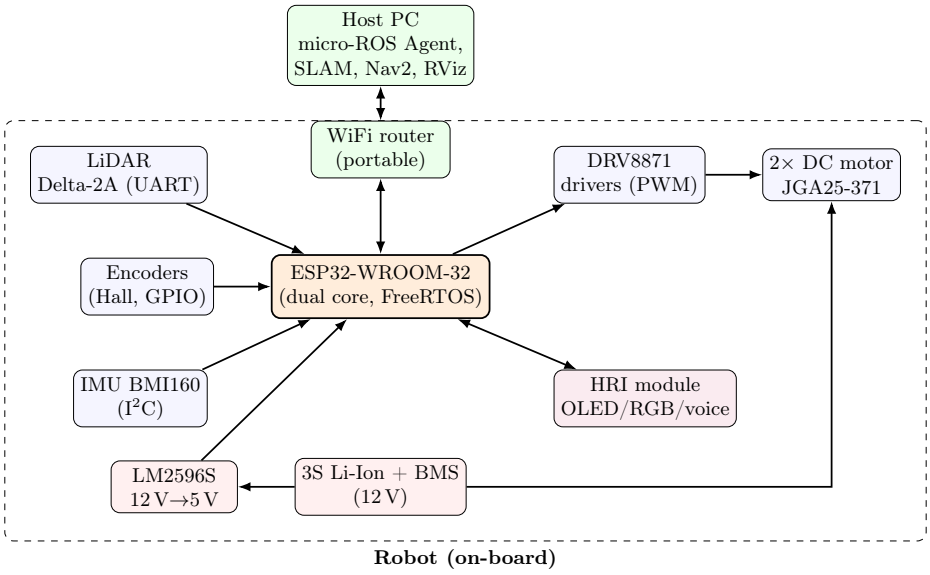


Fig. 3. Hardware architecture: the ESP32 is the sole on-board controller, with power, actuation, perception and communication modules organized around it; the host PC runs SLAM and Nav2 over a private WiFi network.

4 Hardware Architecture

4.1 System overview

Robotito keeps the ESP32-WROOM-32 SoC as the single real-time controller and organizes the rest of the platform around it in four domains – power, actuation, perception and communication (Fig. 3). The microcontroller closes the motor control loop, reads the encoders and the IMU, parses the LiDAR, and exchanges ROS 2 messages with a host PC through a micro-ROS Agent over a private WiFi network.

A laptop runs the heavy computation (SLAM, Nav2, visualization); the robot itself only runs the firmware. This asymmetric split (on-board deterministic control, off-board perception/planning) minimizes cost, albeit introducing a dependency on WiFi link quality (discussed in Sects. 5 and 8).

4.2 Actuation: from steppers to closed-loop DC motors

The NEMA-17 steppers and TMC2208 drivers are replaced by two **JGA25-371 geared DC motors with Hall-effect encoders**, driven by **DRV8871** H-bridge modules. DC motors draw current proportional to load and therefore do not heat at rest, eliminating the thermal problem at its root, while the encoders close a per-wheel velocity PID loop. The wheel speed is obtained from

the encoder ticks as

$$\mathrm{RPM}_{\mathrm{actual}} = \frac{\Delta\mathrm{ticks}}{\text{ticks per rev}} \cdot \frac{60000}{\Delta t_{\mathrm{ms}}}, \tag{1}$$

and a parallel PID controller $u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d\, \dot{e}(t)$ drives the PWM duty cycle to track the commanded speed.

4.3 Differential kinematics and odometry

With $v_r = R\,\omega_d$, $v_l = R\,\omega_i$ ($R = 43\,\mathrm{mm}$, $w = 210\,\mathrm{mm}$), body velocities follow $v_{b,x} = (v_r + v_l)/2$ and $\omega_{b,z} = (v_r - v_l)/w$. Setting $d_C = (d_R + d_L)/2$ and $d\theta = (d_R - d_L)/w$, the pose is integrated by the midpoint rule:

$$x_{t+1} = x_t + d_C \cos\!\Big(\theta_t + \tfrac{d\theta}{2}\Big), \quad y_{t+1} = y_t + d_C \sin\!\Big(\theta_t + \tfrac{d\theta}{2}\Big), \quad \theta_{t+1} = \theta_t + d\theta. \tag{2}$$

Odometry is published together with the `odom`→`base_link` transform following REP-105 [7], completing the URDF kinematic chain to seamlessly integrate with RViz, SLAM Toolbox, and Nav2.

4.4 Power, perception and modular PCB

The unsafe LiPo is replaced by a **3S1P 18650 Li-Ion pack with a protection BMS** in a holder that allows individual cell removal for external charging; an **LM2596S** buck converter splits the supply into a 12 V line for the motors and a clean, isolated 5 V line for the ESP32 and LiDAR, decoupling the logic from motor transients, featuring safe bring-up protection. The **Delta-2A LiDAR is retained** for its compact, symmetric form factor – its defects were purely in firmware (Sect. 6) – and a **BMI160 IMU** (6-DoF, I^2C) is added for yaw estimation and as a substrate for future sensor fusion.

Electronics evolved from a hand-soldered perforated board (**V1**) to a custom 95×80 mm **KiCad** PCB (**V2**). V2 consolidates wiring, reinforces power tracks, integrates the IMU and LiDAR drivers, and provides a 14-pin **expansion header** for the HRI module. The **Fusion 360** chassis uses PLA and TPU, reproducible on standard FDM printers.

5 Embedded Software and Real-Time Architecture

5.1 Firmware structure

The C++/PlatformIO firmware uses three main classes: `JgaMotors` (PID, odometry, `/odom`, `/tf`), `Bmi160Sensor` (calibration, `/imu/data_raw`), and `XiaomiLidar` (parsing, `/scan`). A `main` entry point manages micro-ROS entities and FreeRTOS tasks. This structure ensures readability for educational purposes.

Table 2. FreeRTOS task partitioning across the two ESP32 cores.

Task	Core	Prio.	Function
`SerialReadTask`	0	1	Binary LiDAR parser at 230 400 bps
`RosPublishTask`	1	2	Assemble and publish `LaserScan` on `/scan`
`vTaskMicroROS`	1	3	micro-ROS executor and connection watchdog

5.2 Concurrency under FreeRTOS

To keep the control loop deterministic in spite of WiFi jitter, the firmware exploits the two Xtensa cores at 240 MHz. The high-rate LiDAR byte stream is parsed on **core 0**, while the micro-ROS executor and scan publishing run on **core 1** (Table 2). Pinning the LiDAR parser to its own core guarantees that the UART buffer is drained without interruption, preventing overflow under bursts of incoming bytes. Documenting this standard partitioning is crucial, as it avoids common timing pitfalls (buffer overflow, executor starvation) that often plague microcontroller-based ROS 2 deployments, enhancing reproducibility for educators.

Periodic activity is driven by micro-ROS timers: 33 Hz for motor PID, odometry and `/odom`/`/tf` publication; 20 Hz for IMU reading and the HRI beacon; and 0.2 Hz for clock synchronization.

5.3 Clock synchronization and self-recovery

A recurring lesson of the project was that micro-ROS is sensitive to clock divergence and UDP latency. Every 5 s the firmware invokes the micro-ROS time-sync primitive, which measures the round-trip time to the host and aligns the ESP32 clock with the ROS 2 time base; without this, LiDAR, encoder and IMU timestamps drift and Nav2/AMCL reject the data, breaking the TF tree. The executor also pings the Agent every cycle: after ten consecutive failures it runs an `emergency_stop`, destroys all ROS entities and recreates them, so the robot recovers autonomously from momentary WiFi dropouts. While we report qualitative success (Sect. 7, RNF-5), quantitative communication metrics under degraded WiFi conditions are left for future work.

5.4 Host-side workspace and ROS 2 interface

The ROS 2 host workspace includes `robotito_description` and `robotito_bringup` (launch files, parameters, teleoperation nodes). The robot publishes `/scan`, `/odom`, `/tf`, and `/imu/data_raw`, subscribing to `/cmd_vel` and `/robot_emotion`. A local web application provides virtual joystick and button teleoperation for manual mapping.

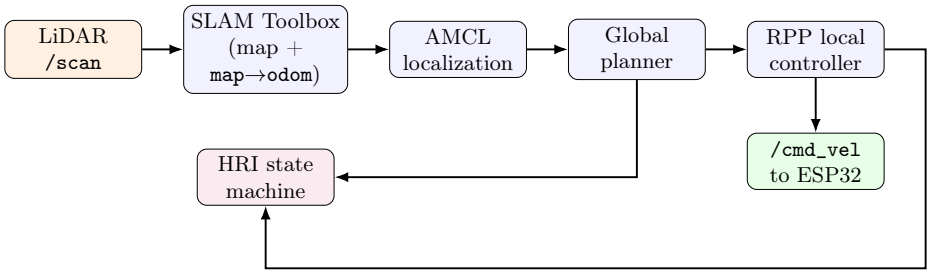


Fig. 4. Navigation pipeline: corrected laser scans feed SLAM Toolbox (mapping) and AMCL (localization); Nav2's global planner and Regulated Pure Pursuit controller produce `/cmd_vel`, while navigation events drive the HRI state machine.

6 Navigation and Human–Robot Interaction

Fixing the LiDAR driver. The two firmware defects of the Delta-2A were corrected at the parser level: the mirror inversion is removed by reversing the distance array before publishing, and the inconsistent start angle is solved with a zero-crossing detector that compares the start angle of consecutive frames to anchor the scan to a fixed reference. Only after this correction does the published `/scan` become geometrically consistent and usable for mapping.

SLAM and autonomous navigation. Mapping uses **SLAM Toolbox** in online asynchronous mode [5]: the operator teleoperates the robot through the web application, the occupancy grid is built in real time, and the map is saved to `.yaml`/`.pgm`. For autonomous operation, **Nav2** loads the saved map: AMCL localizes the robot by aligning LiDAR scans and publishing the `map→odom` correction [4]; a global planner searches a collision-free path, and a *Regulated Pure Pursuit* controller [6] produces smooth velocity commands while a local costmap detects unmapped obstacles and triggers recovery behaviours (Fig. 4). Robotito simply supplies these unmodified algorithms with accurate `/scan` and `/odom` data.

Audiovisual HRI module. An optional module on the expansion header gives Robotito a legible, expressive channel: two OLED "eyes", an RGB "mouth" beacon and a DFPlayer Mini voice synthesizer with a microSD card. The host `Nav2EmotionBridge` maps Nav2 physical thresholds (e.g., velocity, obstacles) to emotional events, publishing to `/robot_emotion` at 5 Hz. The robot's `Robot Emotions` class interprets these via a **twelve-state priority state machine** (e.g., *Happy, Sad, Angry*, plus auxiliary states), that drives the two OLED displays, the RGB LED and the voice output over I^2C, serial and PWM. Higher-priority events (e.g., an obstacle-triggered stop) preempt routine expressions. The module connects and disconnects without modifying the base PCB, demonstrating the expansion architecture in practice.

Table 3. Requirement validation summary. All twelve requirements passed their acceptance test in the domotic environment.

Id	Requirement	Key acceptance result	Status
RF-1	Embedded kinematic control	Straight-line/turn pose error < 2 cm, $< 10°$	✓
RF-2	Native ROS 2 integration	`/scan`, `/odom`, TF stable in RViz	✓
RF-3	Temporal determinism (FreeRTOS)	10 sessions, no resets / TF time jumps	✓
RF-4	Teleoperation & monitoring	Web control + live map, no perceptible lag	✓
RF-5	Autonomous navigation & mapping	SLAM + point-to-point mission with dynamic avoidance	✓
RF-6	Human–robot interaction	12-state machine expressed correctly	✓
RNF-1	Teaching safety	Max. surface temp. 40.1 °C (< 45), no warping	✓
RNF-2	Cost ≤ 200 €	77 € (import) / 145 € (local/EU)	✓
RNF-3	Reproducibility	Full chassis printed on a consumer 3D printer	✓
RNF-4	Modular expansion	HRI module added with no PCB change	✓
RNF-5	Network portability	Full session on a USB-powered private router	✓
RNF-6	Professional finish	Tutor review	✓

7 Experimental Validation

All tests were carried out in the AVISPA laboratory at the University of Málaga, which contains a domotic test apartment that emulates a real home (kitchen, corridors and rooms) and provides everyday static obstacles and the ability to introduce dynamic ones, kept fixed across runs so only the variable under test changed. Twelve requirements – six functional, six non-functional – were each validated by a dedicated test; all twelve passed (Table 3). This *requirements-validation* confirms design compliance rather than benchmarking against alternatives. A supplementary video demonstrates teleoperated mapping and autonomous obstacle avoidance.

Kinematic accuracy (RF-1). From the origin, a continuous command of 0.2 m/s linear and 1 rad/s angular velocity for 5 s was issued on `/cmd_vel`; forward kinematics predicts a final pose of $(-19.2, 14.3)$ cm at $-73.5°$. Over five consecutive repetitions, the maximum planar deviation from the theoretical coordinate stayed **below 2 cm** and the maximum heading error **below 0.18 rad** ($10°$), both within the acceptance thresholds. Encoder odometry alone proved accurate enough for this indoor, low-slip environment, so the IMU and an EKF fusion stage were left out of the final configuration to save computation; they remain available on the expansion header for future fusion work.

Mapping and autonomous missions (RF-5). The robot builds maps of the test apartment without erratic rotations or wall doubling and completes point-to-point missions autonomously: the global planner traces the route, the local controller modulates speed with curvature, and the local costmap detects a deliberately interposed obstacle and re-plans around it without stopping the mission.

Safety, determinism and portability (RNF-1, RF-3, RNF-5). After a 30-minute continuous teleoperation-plus-navigation session, the maximum surface

Table 4. Unit bill of materials by category (two sourcing scenarios).

Category	**Local/EU (€)**	**Import (€)**
Modules (motors, drivers, DC-DC, LiDAR, ESP32, IMU, battery)	115.0	50.5
Support electronics (PCB, headers, passives, switch, cabling)	13.0	10.5
Assembly (PLA, TPU, fasteners, inserts, consumables)	16.5	16.5
Total per robot	**≈144.5**	**≈77.5**

temperature was 40.1 °C at the ESP32 (motors and battery stayed near ambient), 4.9 °C below the 45 °C threshold, and the chassis showed no deformation after months of use. Experiments were performed with *no* microcontroller resets, TF time-jumps or Agent drops, confirming that FreeRTOS core separation isolates the control loop from WiFi fluctuations. A complete session run over a USB-powered portable router (no institutional network) performed identically, demonstrating deployment independence.

Cost and reproducibility (RNF-2, RNF-3). The unit bill of materials is **≈77 €** when sourced from import platforms and **≈145 €** from local/EU retailers (Table 4) – both well under the 200 € target and, consistent with Table 1, roughly one order of magnitude below TurtleBot 3/4. The full chassis prints on a consumer 3D printer (PLA + TPU) with no special equipment.

Open-source release (RNF-3). Hardware, firmware, and configuration files will be released in a public GitHub repository under an open-source licence, enabling independent reproduction of Table 3. Course adoption at the University of Málaga provides near-term testing of this reproducibility.

8 Discussion and Conclusion

We presented *Robotito*, an open-source differential robot that can use Nav2 navigation stack through an ESP32 via micro-ROS, for as little as 77 €. We claim no algorithmic novelty, utilizing existing navigation methods unmodified. Instead, we present an engineering contribution: redesigning a flawed prototype into a reliable system. Educationally, we demonstrate that this affordable platform is safe for unsupervised student use. This distinction between engineering integration and scientific novelty frames our evaluation.

Limitations. We identify five concrete limitations that bound the claims above and define our immediate priorities:

1. **Single environment.** Validation occurred solely in one domotic apartment. Results may not generalize to outdoor spaces or areas with reflective surfaces challenging 2D LiDAR.

2. **No quantitative benchmarks.** A controlled, head-to-head comparison of navigation and localization against platforms like TurtleBot is pending, constituting a priority to formally substantiate its role in teaching fleets.
3. **No systems-level metrics.** Qualitative reliability was observed, but quantitative metrics (CPU load, message latency, packet loss under degraded networks) require formal instrumentation for future extensions.
4. **No classroom study.** While adopted for a new course, student-outcome data is not yet collected. Educational claims currently rest on platform readiness rather than learning impact; assessments are planned.
5. **Hardware constraints.** Odometry currently relies solely on encoders, risking degradation on high-slip surfaces. Additionally, the ESP32-WROOM-32 requires an external JTAG debugger; migration to the ESP32-S3-DevKitC1 is underway.

These limitations do not invalidate the acceptance criteria in Table 3. Instead, they define the boundaries of current claims and outline the evaluation necessary before asserting equivalence to established platforms. Planned extensions include inertial-encoder fusion, a 3-DoF manipulator for mobile-manipulation exercises, fiducial-marker-based (AprilTag) localization, a 12 V battery variant, and multi-robot/swarm experiments – contexts where low per-unit costs make multi-robot laboratory sessions affordable.

Acknowledgments. This work has been supported by grant PID2022-137344OB-C3X, funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Adept MobileRobots: Pioneer 3-DX mobile robot — operations manual. https://www.generationrobots.com (2011), last accessed 2026-06-18
2. Amsters, R., Slaets, P.: Turtlebot 3 as a robotics education platform. In: Robotics in Education (RiE). Advances in Intelligent Systems and Computing, vol. 1023, pp. 170–181. Springer (2019)
3. Belsare, K., Rodriguez, A.C., Sanchez, P.G., et al.: Micro-ROS: The ROS 2 stack for microcontrollers. In: ROSCon / micro-ROS Project (2023), https://micro.ros.org
4. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: Efficient position estimation for mobile robots. In: Proc. of the National Conf. on Artificial Intelligence (AAAI). pp. 343–349 (1999)
5. Macenski, S., Jambrecic, I.: SLAM toolbox: SLAM for the dynamic world. Journal of Open Source Software **6**(61), 2783 (2021)
6. Macenski, S., Singh, S., Martín, F., Ginés, J.: Regulated pure pursuit for robot path tracking. Autonomous Robots **47**(6), 685–694 (2023)
7. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022)

8. Macenski, S., Martín, F., White, R., Ginés Clavero, J.: The marathon 2: A navigation system. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) pp. 2718–2725 (2020)
9. Maier, A., Sharp, A., Vagapov, Y.: Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In: Internet Technologies and Applications (ITA). pp. 143–148 (2017)
10. Mondada, F., Bonani, M., Raemy, X., et al.: The e-puck, a robot designed for education in engineering. In: Proc. of the 9th Conf. on Autonomous Robot Systems and Competitions. vol. 1, pp. 59–65 (2009)
11. Quigley, M., Conley, K., Gerkey, B., et al.: ROS: An open-source robot operating system. In: ICRA Workshop on Open Source Software. vol. 3, pp. 1–6 (2009)
12. Thrun, S., Montemerlo, M., Dahlkamp, H., et al.: Stanley: The robot that won the DARPA grand challenge. Journal of Field Robotics **23**(9), 661–692 (2006)
13. Wurman, P.R., D'Andrea, R., Mountz, M.: Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI Magazine **29**(1), 9–20 (2008)

Reviving a Legacy Pioneer 2-AT: An Open-Source Retrofit for Outdoor GNSS Navigation with ROS 2 and Nav2

Claudia González Mena[1 [0009−0009−6067−4027]], Alberto Tudela[1 [0000−0001−6796−5286]], Camilo A. Ruiz-Beltrán[1 [0000−0001−8270−2062]], José Galeas[1 [0009−0007−9978−8406]], and Antonio Bandera[1 [0000−0003−3147−0307]]

Dpto. de Tecnología Electrónica, E.T.S. de Ingeniería de Telecomunicación, Universidad de Málaga, Málaga, Spain
claudiagnzmena,ajtudela,camilo,jgaleas,ajbandera@uma.es

Abstract. Research laboratories accumulate ageing mobile robots that stay mechanically sound but become unusable once their proprietary controllers and software fall out of support. We present a complete, open-source retrofit of a *Pioneer 2-AT*—a skid-steer all-terrain platform from the early 2000s—turning it into a modern ROS 2 robot for autonomous outdoor GNSS navigation. The lead-acid power system is replaced by a custom Battery Management System (BMS) PCB that safely operates a LiPo pack and delivers regulated 12 V and 19 V rails; the obsolete controller is superseded by an Intel NUC running ROS 2 and Nav2; and a 16-beam 3D LiDAR, a multi-constellation GNSS receiver and an IMU are added as exteroceptive sensors. The GNSS and IMU are exposed as *native* ROS 2 nodes through micro-ROS on an ESP32, decoupling acquisition from the host. A dual extended-Kalman-filter pipeline with a `navsat_transform` stage fuses wheel odometry, inertial data and GNSS fixes, enabling latitude/longitude waypoint missions. We detail the hardware, electronic and software re-engineering and report indoor mapping, GNSS fix characterization and a tethered validation of the GPS waypoint pipeline. All firmware, PCB designs and configuration are released open-source.

Keywords: Legacy robot retrofit · Outdoor navigation · ROS 2 · Nav2 · micro-ROS · GNSS · Battery management system · Sensor fusion.

1 Introduction

Mobile robotics laboratories rarely retire robots because the hardware fails; more often a mechanically robust platform becomes a museum piece once its embedded controller, toolchain or host OS is no longer maintained, and porting modern software to a closed architecture is prohibitive [4]. The *Pioneer* family of ActivMedia/MobileRobots robots is paradigmatic: thousands sold to universities in the late 1990s–2000s remain mechanically excellent, yet their original

ARIA/P2OS software stack and bulky lead-acid batteries make them increasingly hard to use [1, 2]—and discarding them is wasteful, amid growing concern over electronic waste [8].

Meanwhile, the software substrate has consolidated: ROS [16] has matured into ROS 2, the *de facto* research standard built on real-time-capable DDS middleware [11], whose Nav2 navigation stack offers a behaviour-tree architecture with pluggable planners and controllers validated in long autonomous missions [12, 10]. On the embedded side, micro-ROS pushes the same middleware onto 32-bit microcontrollers, so an ESP32 can join the ROS 2 graph as a first-class node [3, 13]—making it feasible to give a legacy chassis a complete second life rather than replacing it.

Most academic robotics, moreover, is validated *indoors*, where a 2D LiDAR and grid-based SLAM suffice. Outdoor operation is qualitatively different—unbounded, without reliable wall-like features for scan matching, and demanding a globally referenced position estimate, typically from fusing wheel odometry and inertial data with a Global Navigation Satellite System (GNSS) receiver so the robot can be sent to geographic goals [6, 7]. The all-terrain Pioneer 2-AT, with four driven wheels and high ground clearance, suits this regime—once modernized.

Starting from a non-operational Pioneer 2-AT, we perform a full, documented and open-source retrofit and bring it to autonomous outdoor GNSS navigation, validated indoors and through a tethered test of the GPS waypoint pipeline. The contributions are: (i) a **power-system redesign** around a custom BMS PCB that replaces the lead-acid pack with a lighter LiPo battery, adding per-cell protection, balancing and regulated 12 V/19 V rails (Sect. 4); (ii) a **perception and mechanical upgrade** adding a 16-beam 3D LiDAR, a multi-constellation GNSS receiver and a 6-DoF IMU on printed mounts within a REP-105 transform tree (Sect. 5); (iii) a **micro-ROS embedded layer and software stack** exposing the GNSS and IMU as native ROS 2 nodes from an ESP32 and wrapping the legacy ARIA/P2OS controller in a modular ROS 2 driver coupled to Nav2 and a dual-EKF `robot_localization`/`navsat_transform` pipeline (Sects. 6–8); and (iv) an **experimental validation** of indoor SLAM, GNSS characterization and the tethered GPS waypoint pipeline (Sect. 9).

2 Related Work

Legacy-robot retrofit, ROS 2 and micro-ROS. Extending the service life of robotic hardware by replacing only its electronics and software is an established but under-reported practice [8]; the Pioneer platforms in particular have been repeatedly re-controlled, with community efforts exposing the P2OS serial protocol to ROS 1 and kits replacing their batteries. We follow this bridging philosophy—keeping the P2OS firmware and reusing the ARIA library [2] behind a thin ROS 2 wrapper (Sect. 7)—but go beyond the bridge: we redesign the power chain, add an exteroceptive sensor suite and target the *outdoor* autonomy the indoor-

Fig. 1. Before/after of the retrofit. *Left:* the original Pioneer 2-AT. *Right:* the retrofitted robot.

oriented robot was never built for, on the modern ROS 2/DDS, Nav2 [12, 10] and micro-ROS [3, 13] substrate above.

GNSS-based outdoor navigation and SLAM. Outdoor mobile robots localize by fusing proprioceptive sensors with absolute GNSS measurements [6, 7]. The canonical ROS solution chains two extended Kalman filters from `robot_localization` [15] with a `navsat_transform` node and a Nav2 GPS waypoint follower on top (detailed in Sect. 8). We instantiate and tune this pipeline on a retrofitted skid-steer platform, surfacing practical issues (IMU yaw referencing, multi-constellation fix status, 3D-to-2D LiDAR reduction) seldom documented end-to-end. For indoor bring-up we rely on 2D LiDAR SLAM, mature through SLAM Toolbox [9], as a controlled baseline before the GNSS-referenced outdoor regime.

3 Baseline Platform and Limitations

The *Pioneer 2-AT* (Fig. 1, left) is a four-wheel skid-steer research robot. Its differential-like drive couples the two wheels on each side, so heading is changed by a velocity difference between the left and right banks. The original platform is built around two reversible DC gear-motors with high-resolution encoders, an embedded microcontroller running the P2OS firmware, and three 12 V 7 Ah sealed lead-acid batteries in parallel, historically programmed through the ARIA C++ library over an RS-232 link to a tethered or on-board laptop [1, 2].

While the mechanics and drivetrain are in excellent condition, four factors made the platform unusable for current research: obsolete, unmaintained software (P2OS/ARIA, with no ROS 2 interface); a heavy, degraded lead-acid power system that cannot supply the regulated 19 V rail a modern computer needs; no exteroceptive sensing beyond wheel odometry and sonar; and no outdoor localization. These map directly onto the retrofit requirements—a regulated power system, a modern ROS 2 stack, a perception suite and a globally referenced navigation pipeline—yielding the robot of Fig. 1 (right).

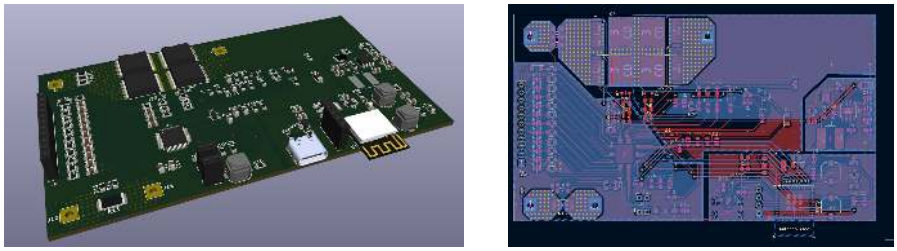


Fig. 2. Custom multi-cell BMS PCB designed in KiCad. *Left:* 3D render with per-cell balancing, protection and the regulated rails. *Right:* routed two-layer layout, with DRC driven to zero errors before fabrication.

4 Power Subsystem: a Custom BMS

Replacing the lead-acid bank with a lithium-polymer (LiPo) pack roughly halves the energy-storage mass and removes the degraded cells, but LiPo chemistry is unforgiving: cells must never be over-charged, over-discharged or allowed to drift out of balance [17, 19], so a Battery Management System (BMS) is mandatory. The off-the-shelf alternative—a generic protection board plus separate DC/DC bricks—would have been bulky, poorly fitted to the Pioneer's internal bay and unable to expose battery state to ROS 2. We instead designed a dedicated BMS as a single two-layer PCB in KiCad (Fig. 2).

The board manages a series LiPo pack (10 cells, 37 V fully charged) and performs three functions: *protection*, disconnecting the pack on over-voltage, under-voltage, over-current or short circuit; *passive balancing*, equalizing per-cell voltages during charge so the pack ages uniformly; and *power conditioning*, deriving two regulated rails—a 12 V line for the Pioneer base and a 19 V line for the NUC and LiDAR (the ESP32 micro-ROS boards draw USB power from the NUC). For a pack of nominal cell voltage V_{cell} in an n_s-series configuration delivering payload power P, the pack current is

$$I_{\text{pack}} = \frac{P}{\eta\, n_s\, V_{\text{cell}}}, \tag{1}$$

where η is the aggregate regulation efficiency. With $n_s V_{\text{cell}} \approx 36\,\text{V}$ and a measured payload $P \approx 102\,\text{W}$ (NUC, LiDAR and sensors), Eq. (1) gives a continuous draw of $\approx 3.06\,\text{A}$ at $\eta = 0.9$, comfortably within the protection threshold, and a usable runtime $t \approx \eta\, n_s V_{\text{cell}} Q/P \approx 3.26\,\text{h}$ for a 10 Ah pack. The board was validated at design level—routing took several iterations, with the DRC driven from dozens of violations to zero before release—and left ready for fabrication; in the campaign reported here the robot was powered from a bench supply emulating the BMS rails, so the rest of the stack could be exercised in parallel with PCB manufacturing.

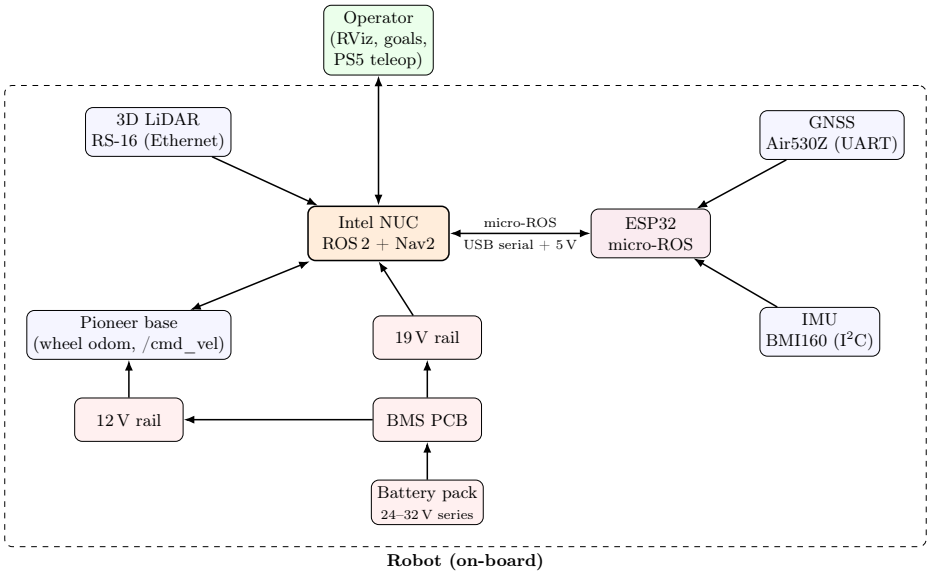


Fig. 3. System architecture of the retrofitted robot.

5 Mechanical and Perception Retrofit

The retrofit keeps the Pioneer drivetrain and adds a modern compute-and-sensing payload organized in four domains—power, compute, perception and embedded sensing—around an Intel NUC host (Fig. 3). The NUC runs ROS 2 and Nav2; perception is provided by a **RoboSense RS-LiDAR-16**, a spinning LiDAR with 16 beams and a full 360° horizontal field of view [18]; and absolute-localization sensing (GNSS and IMU) is delegated to an ESP32 micro-ROS node. The LiDAR sits on a 3D-printed mast, designed in Fusion 360 to bolt onto the existing top-plate hole pattern, raising it above the deck for an unobstructed horizon. It communicates over Ethernet, its SDK streaming the point cloud into ROS 2 once the NUC and LiDAR share an IP subnet (Fig. 5).

The GNSS receiver and its antenna are housed in a purpose-designed, 3D-printed enclosure (Fig. 4) that protects the electronics while leaving the patch antenna a clear sky view, with a removable lid and a strain-relieved cable exit; it took several print iterations to correct antenna-window size, wall thickness and cable routing.

6 Embedded Sensing with micro-ROS

Rather than wiring the GNSS and IMU as raw serial peripherals of the NUC, we host their drivers on an **ESP32-WROOM** and expose each as a native ROS 2 node through **micro-ROS** [3]. This decouples timing-sensitive acquisition

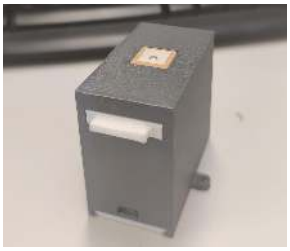


Fig. 4. The printed enclosure housing of the Air530Z GNSS.

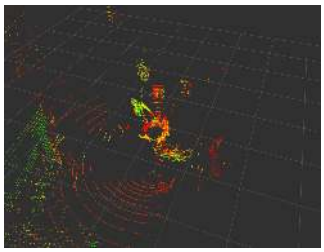


Fig. 5. A point cloud from the RS-LiDAR-16.

from the host, keeps the sensor heads close to their antennas over a single USB link, and yields standard ROS 2 messages (`sensor_msgs/NavSatFix`, `sensor_msgs/Imu`) the localization stack consumes directly. The firmware is C++ with PlatformIO over FreeRTOS.

The GNSS receiver is a **Grove Air530Z**, a multi-constellation module tracking GPS, BeiDou, GLONASS, Galileo, QZSS and SBAS over NMEA/UART. A customized `TinyGPS++` parses these, reporting—beyond latitude, longitude and altitude—the active constellation set and fix status, exposed as a lock flag in the `status` field of `NavSatFix` ("fixed"/"no fix"); this matters because a cold start needs time to acquire satellites and the downstream EKF must ignore fixes until the lock is valid. Orientation and angular rate come from a **Bosch BMI160**, a 6-DoF unit read over I^2C, whose node publishes `sensor_msgs/Imu` and supplies the heading reference the global EKF needs—without it, GNSS alone cannot disambiguate yaw at low speed [5]. Finally, a micro-ROS *agent* bridges the ESP32 to the DDS graph from a **Docker** container (avoiding a full toolchain on the NUC), connecting over serial on the powering USB cable and time-synchronizing on connection so sensor timestamps share the ROS 2 time base—a prerequisite for the filters and TF tree to accept the data.

7 Software Migration and Transform Tree

The open-source, modular `pioneer_ros2` stack[1] keeps the original P2OS firmware on the base and reuses the ARIA library to speak its serial protocol, exposing only the minimum needed to drive the robot as a ROS 2 node. Layered into a core, an ARIA driver node and message packages, the driver subscribes to `geometry_msgs/Twist` on `/cmd_vel` and publishes wheel odometry as `nav_msgs/Odometry` on `/odom` with the `odom`→`base_link` transform, restoring the basic "drive and report" contract expected by ROS 2.

With its left and right banks mechanically coupled, the platform is modelled as a differential drive with effective track width b. From the per-side wheel displacements d_L and d_R over an interval, the body displacement and heading

[1] `https://github.com/grupo-avispa/pioneer_ros2`

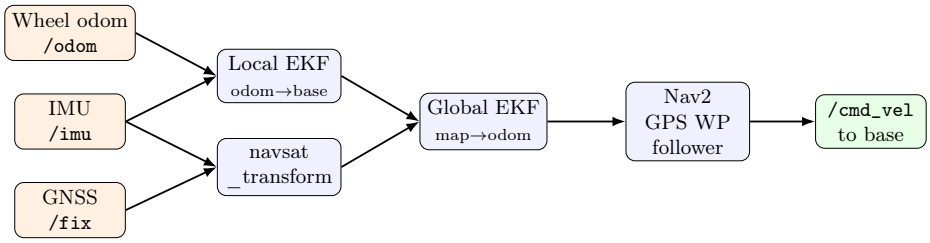


Fig. 6. Outdoor localization and navigation pipeline. A local EKF fuses odometry and IMU; `navsat_transform` projects GNSS fixes into the local Cartesian frame; a global EKF fuses everything into `map`→`odom`; and the Nav2 GPS waypoint follower converts latitude/longitude goals into `/cmd_vel` commands.

change are $d_C = (d_R + d_L)/2$ and $d\theta = (d_R - d_L)/b$, and the pose is integrated by the midpoint rule

$$x_{t+1} = x_t + d_C \cos\Big(\theta_t + \tfrac{d\theta}{2}\Big), \quad y_{t+1} = y_t + d_C \sin\Big(\theta_t + \tfrac{d\theta}{2}\Big), \quad \theta_{t+1} = \theta_t + d\theta. \tag{2}$$

Skid steering violates the rolling-without-slipping assumption during turns, so b is calibrated empirically to absorb part of the slip, and the IMU yaw rate is later fused to correct the residual heading drift.

SLAM Toolbox and the Nav2 costmaps used for the indoor baseline expect a planar `sensor_msgs/LaserScan`, but the RS-LiDAR-16 delivers a 16-layer `PointCloud2` (Fig. 5); an intermediate `pointcloud_to_laserscan` stage therefore collapses the layers within a height band into a single virtual scan ring, preserving the obstacle silhouette needed for 2D mapping and local avoidance while deferring full 3D perception to future work. Outdoors, the same cloud also feeds the local costmap as a 3D obstacle source. All frames follow REP-105 [14]: the local EKF produces the continuous `odom`→`base_link` transform; the global EKF and `navsat_transform` produce `map`→`odom`; and static transforms place the LiDAR, GNSS antenna and IMU relative to `base_link`, with time synchronization across the NUC, LiDAR and micro-ROS sensors keeping the tree consistent.

8 Outdoor GNSS Navigation

Outdoor localization follows the established `robot_localization` dual filter pattern [15] (Fig. 6). A *local* EKF fuses wheel odometry (Eq. (2)) and the IMU into a smooth, drift-bounded but only locally consistent `odom`→`base_link` estimate, while a *global* EKF adds the GNSS-derived position to estimate `map`→`odom`, anchoring the pose to an absolute frame.

The `navsat_transform` node converts each `NavSatFix` (latitude φ, longitude λ in WGS84) into the local `map` frame: the fix is projected to UTM planar coordinates (E, N) and, given the datum (E_0, N_0) at the map origin and the

Table 1. Retrofit requirement validation summary.

Id	Requirement	Key acceptance result	Status
R-1	ROS 2 base control	`/cmd_vel` drive + `/odom`/TF restored, PS5 teleop	✓
R-2	3D LiDAR perception	16-layer cloud streamed; 2D scan for SLAM/costmap	✓
R-3	micro-ROS sensing	GNSS & IMU as native ROS 2 nodes via ESP32	✓
R-4	GNSS fix & status	Multi-constellation fix with reported lock flag	✓
R-5	Indoor SLAM map	Consistent occupancy grid (SLAM Toolbox)	✓
R-6	GPS waypoint pipeline	Lat/lon goals projected and tracked, robot tethered	(✓)
R-7	Custom BMS (design)	Series-pack protection/balancing, 12 V/19 V rails, DRC clean	(✓)

yaw offset ψ_0 between the UTM grid and the robot's initial heading (from the IMU), the `map`-frame position is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\psi_0 & -\sin\psi_0 \\ \sin\psi_0 & \cos\psi_0 \end{bmatrix} \begin{bmatrix} E - E_0 \\ N - N_0 \end{bmatrix}. \tag{3}$$

The same transform runs in reverse to turn geographic waypoints into metric goals; a correct ψ_0 is critical, since an erroneous initial yaw rotates the whole path—hence the reliable IMU heading reference is a hard design requirement. Geographic missions run on the **Nav2 GPS waypoint follower**, which converts each latitude/longitude waypoint to a `map`-frame goal through Eq. (3) and sequences them through the Nav2 behaviour tree [12]: the global planner traces a path, a Regulated Pure Pursuit controller [10] produces smooth velocities suited to long, gently curving outdoor paths, and the LiDAR-fed local costmap handles unmapped obstacles, advancing to the next waypoint on arrival within tolerance.

9 Experimental Validation

Bring-up and verification were carried out indoors in our laboratory, with the robot powered from a bench supply emulating the BMS rails (Sect. 4) while the PCB completed fabrication. Because the power electronics were not yet on board, the GPS waypoint-following pipeline was exercised with the robot tethered—driven from real GNSS fixes and geographic goals near the laboratory—rather than in a free outdoor run; a full autonomous outdoor mission on board power remains the immediate next step. Table 1 summarizes the requirement validation.

After migrating to `pioneer_ros2` (R-1), the robot accepted velocity commands and reported wheel odometry and the `odom`→`base_link` transform consistently in RViz, with the PS5 gamepad giving reliable teleoperation for SLAM and manual recovery. The micro-ROS GNSS node (R-3, R-4) acquired a multi-constellation fix after a ≈ 120 s cold start, tracking 9–12 satellites and reporting

the lock through the `NavSatFix` status field ; the explicit flag let the global EKF reject pre-lock fixes that would otherwise corrupt `map`→`odom`. From the 2D scan reduced from the RS-LiDAR-16 (R-2, R-5), SLAM Toolbox built a consistent occupancy grid of the laboratory [9], confirming the odometry of Eq. (2), the LiDAR pipeline and the transform tree were correctly integrated.

For the GPS navigation pipeline (R-6), the dual-EKF chain produced a stable globally referenced pose, and—with the robot tethered—the Nav2 GPS waypoint follower correctly projected latitude/longitude goals near the laboratory into `map`-frame targets through Eq. (3) and issued consistent velocity commands toward them. This exercises the end-to-end waypoint logic (`navsat_transform` projection, dual-EKF fusion and behaviour-tree sequencing) short of a free outdoor drive, which awaits the on-board BMS. The BMS (R-7) was validated at design level (schematic capture, two-layer routing, DRC driven to zero errors; Fig. 2) The campaign thus validates the retrofit through indoor operation and a tethered run of the GPS pipeline; the two remaining open items—fabricating the BMS and conducting a free outdoor mission on board power—are the main limitations and the immediate next step.

10 Conclusion

We presented a complete, open-source retrofit that returns a legacy Pioneer 2-AT to active service as a modern ROS 2 robot capable of autonomous outdoor GNSS navigation, spanning every layer of the platform: a custom BMS PCB, a 3D-LiDAR/ GNSS/IMU perception upgrade, a micro-ROS embedded layer on an ESP32, and a software stack that reuses the legacy Aria/P2OS controller through a modular ROS 2 driver coupled to Nav2 with a dual-EKF and `navsat_transform` pipeline. Indoor SLAM, GNSS characterization and a tethered test of the GPS waypoint pipeline validate the design short of a free outdoor mission. Beyond the specific platform, the project illustrates a reproducible recipe for extending the service life of obsolete research robots—an attractive alternative to retirement on both economic and environmental grounds [8]. Future work will close the power loop with the fabricated BMS, add RTK-augmented GNSS for centimetre-level accuracy, and exploit the LiDAR in full 3D for terrain-aware navigation. All firmware, PCB designs, mounts and configuration are released open-source.

Acknowledgments. This work has been supported by grant PID2022-137344OB-C3X, funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. ActivMedia Robotics: Pioneer 2 / PeopleBot operations manual. ActivMedia Robotics, LLC (2003), last accessed 2026-06-18

2. Adept MobileRobots: ARIA: Advanced robot interface for applications — reference manual. https://github.com/reedhedges/AriaCoda (2021), last accessed 2026-06-18
3. Belsare, K., Rodriguez, A.C., Sanchez, P.G., et al.: Micro-ROS: The ROS 2 stack for microcontrollers. In: ROSCon / micro-ROS Project (2023), https://micro.ros.org
4. Cebollada, S., Payá, L., Flores, M., Peidró, A., Reinoso, O.: A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data. Expert Systems with Applications **114**, 195–223 (2021)
5. Forster, C., Carlone, L., Dellaert, F., Scaramuzza, D.: On-manifold preintegration for real-time visual–inertial odometry. IEEE Transactions on Robotics **33**(1), 1–21 (2017)
6. Groves, P.D.: Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems. Artech House, 2nd edn. (2013)
7. Kassas, Z.M., Closas, P., Gross, J.: Navigation systems panel report: Navigation systems for autonomous and semi-autonomous vehicles. IEEE Aerospace and Electronic Systems Magazine **34**(5), 82–84 (2019)
8. Lu, B., Liu, J., Yang, J., Li, B.: The environmental impact of technology innovation on WEEE management by multi-life-cycle assessment. Journal of Cleaner Production **89**, 148–158 (2015)
9. Macenski, S., Jambrecic, I.: SLAM toolbox: SLAM for the dynamic world. Journal of Open Source Software **6**(61), 2783 (2021)
10. Macenski, S., Singh, S., Martín, F., Ginés, J.: Regulated pure pursuit for robot path tracking. Autonomous Robots **47**(6), 685–694 (2023)
11. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022)
12. Macenski, S., Martín, F., White, R., Ginés Clavero, J.: The marathon 2: A navigation system. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) pp. 2718–2725 (2020)
13. Maier, A., Sharp, A., Vagapov, Y.: Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In: Internet Technologies and Applications (ITA). pp. 143–148 (2017)
14. Meeussen, W.: REP 105: Coordinate frames for mobile platforms. https://www.ros.org/reps/rep-0105.html (2010), last accessed 2026-06-18
15. Moore, T., Stouch, D.: A generalized extended Kalman filter implementation for the Robot Operating System. Intelligent Autonomous Systems 13 (IAS-13) **302**, 335–348 (2016)
16. Quigley, M., Conley, K., Gerkey, B., et al.: ROS: An open-source robot operating system. In: ICRA Workshop on Open Source Software. vol. 3, pp. 1–6 (2009)
17. Rahimi-Eichi, H., Ojha, U., Baronti, F., Chow, M.Y.: Battery management system: An overview of its application in the smart grid and electric vehicles. IEEE Industrial Electronics Magazine **7**(2), 4–16 (2013)
18. RoboSense: RS-LiDAR-16: 16-beam mechanical LiDAR — user guide and SDK. https://github.com/RoboSense-LiDAR/rslidar_sdk (2023), last accessed 2026-06-18
19. Xiong, R., Li, L., Tian, J.: Towards a smarter battery management system: A critical review on battery state of health monitoring methods. Journal of Power Sources **405**, 18–29 (2018)

Educational robotics beyond ROS: a custom library for direct transfer of Webots controllers in the Turtlebot3

Jaime P. Pérez-Moro[1][0009−0002−3143−2386], Alejandro Romero[1][0000−0002−0507−8320], Hector Quintian[2][0000−0002−0268−7999], and Francisco Bellas[1][0000−0001−6043−1468]

[1] Integrated Group for Engineering Research (GII), CITIC research center, Universidade da Coruña, A Coruña, Spain
{j.p.perez,alejandro.romero.montero, francisco.bellas}@udc.es
[2] Grupo de investigación Ciencia y Técnica Cibernética (CTC), CITIC research center, Universidade da Coruña, A Coruña, Spain
hector.quintian@udc.es

Abstract. Robot Operating System (ROS) is becoming a standard in robotics courses at higher education, but its installation, configuration and distributed programming model can introduce a significant entry barrier in the new landscape of AI-based master and degree studies, where many students do not have a strong informatics background. This paper presents an educational experience that enables students to program a TurtleBot3 Burger directly in Python, while preserving a direct transfer path from simulation to the physical robot. The approach is based on a custom library that mirrors the Webots API, allowing the same controller code to be executed in the simulator and on the real TurtleBot3 with minimal modification. The library provides access to the robot motors, wheel encoders, inertial sensors and LiDAR, and is complemented by scripts that automate deployment and execution on the physical platform. It was used in a master-level course on introductory autonomous vehicles, and initial student feedback suggests that the approach reduces the perceived complexity of ROS-related infrastructure and helps learners focus on algorithmic logic and real-world behaviour.

Keywords: Educational robotics · Simulation-to-reality transfer · ROS

1 Introduction

The use of ROS [1] in technical subjects and courses related with autonomous robotics at higher education is becoming a standard [2][3]. Its open-access nature, hardware abstraction and huge community of contributors make ROS as the optimal choice for teachers that aim to provide students with the most solid and long-term training in robotics. In addition, the current offer of affordable and versatile real platforms for educational/research purposes is mainly ROS-compatible and even ROS-exclusive, like those from Unitree, Universal Robots, PAL or Clearpath Robotics.

Since the "explosion" of generative AI in 2022, the interest of students in titles related with Artificial Intelligence (AI) has notably increased and, consequently, the offer of new titles related with this area has raised [4]. Many of these new students and titles do not have or require the high technical background associated with ROS, although they need learning about autonomous robotics in a solid and sound way. There are notably relevant initiatives that aim to "smooth" the ROS learning curve, such as the web-based Unibotics framework [5][6], but the real situation in many of these new courses is that installing Linux or relying on virtual machines is complicated, mainly due to time constraints in the subjects, and the lack of support from the institutions.

With this background, the current paper describes a custom library developed at the University of Coruña (UDC), in Spain, together with the educational experience in which it was applied. The library was created for a technical master program named "Industrial Informatics and Robotics" [7]. This subject is a clear example of the situation presented above, as it is mainly technical, but the number of students attending it with limited programming and computer science background increases every year. In particular, the experience described here was carried out in the realm of a subject named "Introductory Autonomous Vehicles", where students learn the fundamentals of this field, including topics like perception, mapping, localization and control.

The practical part of the subject implies programming the response of an autonomous robot, in simulation and in the real-world, requiring a minimum proficiency level as a computer scientist. The specific platform existing in the UDC for this subject is the TurtleBot3 Burger, a ROS-exclusive platform that is shared with other subjects in the same master. To provide this heterogeneous group of students with a sound and simple educational environment that allow them to practice the subject fundamentals without relying on ROS, we decided to develop a custom library so they can program it in a simpler way using the Webots simulation environment [8].

This work describes the implementation details of this library, how it was validated in the simulator and the real robot, and its application during academic year 2025/26 within the aforementioned subject.

2 Custom library development

This section describes the development of the custom library. First, the Webots simulation environment is presented, justifying its selection for this work. Next, the design and implementation of the library are detailed, covering the low-level access to the TurtleBot3 Burger hardware and the way its interface mirrors the Webots API so that the same controller runs unchanged in simulation and on the real robot. Finally, the operational validation carried out before using the library in a teaching setting is reported.

2.1 Webots simulation environment

Webots [8] is an open-source and widely adopted simulation platform that was selected for this course after evaluating several of the alternatives commonly used in educational robotics [9], based on a combination of features well-suited to an educational context with a heterogeneous student background. Its physics engine, the Open Dynamics Engine (ODE), enables realistic simulations supporting collisions, friction and gravity, which are essential for an accurate reproduction of robot behaviour. It is also flexible in terms of programming languages, since controllers can be programmed in Python, C, C++, Java or MATLAB, a key advantage for groups with limited programming background.

Webots further supports many popular robot models out of the box, including the TurtleBot3 Burger, which avoids building models from scratch and makes experimentation with the target platform straightforward. World creation, controller programming and debugging are all carried out from a single unified interface, and its extensive documentation and active community lower the entry barrier for new users. The accuracy of motion simulation and the comparison with other simulators that supported this choice are discussed in [10].

2.2 Design and implementation

The development started by addressing low-level access to the TurtleBot3 Burger, controlling its components directly from Python and without relying on ROS. The robot integrates a Raspberry Pi 3 as the main computer, two Dynamixel XL430-W250 servomotors driving the wheels, an OpenCR 1.0 board hosting an inertial measurement unit (IMU), and an LDS-01 LiDAR, Fig. 1 illustrates how all of these components are connected together.

The servomotors connect to the Raspberry Pi 3 through the OpenCR 1.0 board over a serial port. As the board ships with a ROS-oriented firmware, it was reprogrammed using the Arduino IDE with custom code based on the official Dynamixel2Arduino library [11]. This firmware runs a main loop that monitors the serial input for commands from the Raspberry Pi 3 and reacts to two types of request: setting the velocity of a wheel, and reading a wheel encoder value, which is returned over the serial link.

Communication on the Raspberry Pi side was handled with the `pyserial` library. Access to the IMU followed a similar approach, extending the firmware with reading functions adapted from the official OpenCR examples to expose raw accelerometer and gyroscope data and the computed roll, pitch and yaw values. The LiDAR LDS-01, by contrast, is connected through an independent USB2LDS board that did not require reprogramming, and its data are read directly from the assigned serial port by a dedicated class that performs the acquisition in a separate thread.

These low-level components were then unified into a single library whose key design decision was to mirror the Webots API [12] as closely as possible, so that the same controller code could run unchanged both in simulation and real robot. To this end, the library functions were given the same names, behaviours and

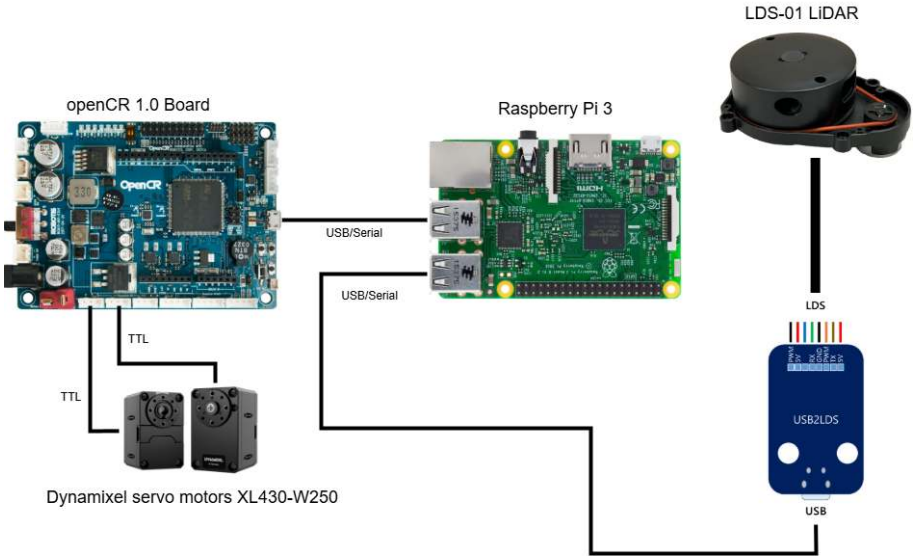


Fig. 1. TB3 Burger connection diagram.

return values as their Webots counterparts, following a modular, object-oriented architecture in which each component is encapsulated in its own class.

A `Robot` class acts as the main entry point, centralising access to the devices and exposing a `step(timestep)` method that emulates the Webots simulation timestep so that sensor readings and command execution occur at consistent time intervals on the physical robot. The remaining classes (`Motor`, `PositionSensor`, `Accelerometer`, `Gyro`, `InertialUnit` and `Lidar`) follow the same principle and replicate the behaviour of their Webots equivalents, for instance converting raw encoder counts into radians as Webots does. In practice, a small code block at the beginning of any Webots controller checks whether the Webots `Robot` class is available; if it is not, indicating execution on the physical hardware, the custom library is imported instead, so no further modification of the controller is needed as seen in Fig. 2.

Finally, the transfer and execution of code on the real robot were automated through a set of scripts working on both Linux and Windows, removing the need to manually establish an SSH connection and copy files. The scripts rely on `scp` and `ssh`, and offer three commands: *deploy*, which transfers the Python controller to the TurtleBot3 Burger given its IP address; *run*, which transfers and executes the code on the robot; and *clean*, which removes the transferred file. The library, scripts and documentation are openly available in our GitHub repository.[3]

[3] TurtleBot3-NoROS-Library: https://github.com/jpperezm/TurtleBot3-NoROS-Library.

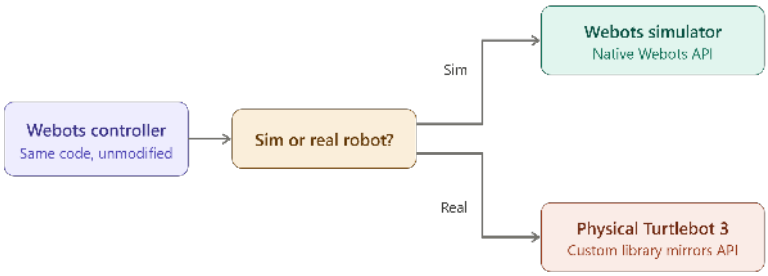


Fig. 2. Execution flow of a Webots controller.

For reproducibility, the software stack used in the reported experience is the following: Webots R2025a on the students' computers, running either Windows or Linux; Ubuntu Server 22.04 LTS on the Raspberry Pi 3 of the TurtleBot3, with Python 3.10 and `pyserial` 3.5; and Arduino IDE 2.3 together with Dynamixel2Arduino 0.8.1 for building the OpenCR 1.0 firmware.

2.3 Operational validation

Before relying on the library in a teaching setting, each component was checked individually, without a control loop, to confirm that the same controller produced equivalent results in simulation and on the physical robot. Three checks were performed: moving the robot 25 cm forward using the wheel encoders, commanding a 90-degree left turn governed by the gyroscope, and advancing towards an obstacle until a target LiDAR reading was reached. In every case the simulated and real runs were close and, importantly, highly repeatable, which is what matters for transferring a controller between the two. The aim was not to characterise sensor error precisely, but to confirm that each component behaves consistently enough for code developed in simulation to carry over to the real robot.

All these tests were confirmed by a final experiment that combined all the components in a single task: the robot had to move 25 cm forward using the encoders, turn 45 degrees to the left using the gyroscope, and then advance while avoiding obstacles with the LiDAR, reacting with a 90-degree turn when an obstacle was detected within 20 cm. The controller was first developed and tuned in Webots, the laboratory was arranged to match the simulated scenario, and the same code was then transferred to the physical TurtleBot3 using the automated scripts without any modification. The robot reproduced the programmed sequence reliably on both platforms[4], confirming that controllers written for

[4] A video comparing the execution of the same controller in simulation and on the real robot is available at https://youtu.be/bzEeayc3Oj4.

Webots can be transferred directly to the real robot and that the Webots–TurtleBot3 combination is a viable solution for teaching robotics without ROS.

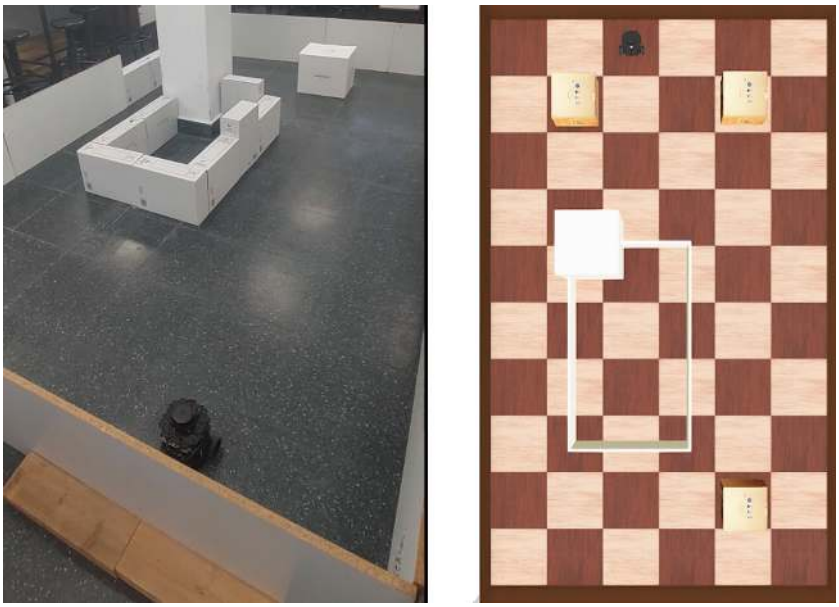


Fig. 3. Real and simulated environment used for the activity.

3 Results

This section describes the educational experience that was carried out in the scope of the "Introductory Autonomous Vehicles" subject from the "Industrial Informatics and Robotics" master during course 2025/26, where the custom library was used. Six students were enrolled in the course this year, as it is not an elective subject, so the master's students are distributed across different courses. The practical task they had to solve was the following:

- Design of a deliberative autonomous architecture, so the TurtleBot3 Burger robot can navigate from one point to another specified by the user, avoiding obstacles and following the shortest path. To this end, the architecture must solve 3 independent problems: (1) map creation (2) path planning (3) navigation. It must be solved in Webots and also in the real robot.
- Teachers provide 3 different environments in simulation, and setup one of them in the real world. They are created as grids, with known maximum dimensions, 6×10 cells, with each cell measuring 40×40 cm. The robot starts from a base cell located next to a wall, but its absolute position and initial

orientation are not known in advance, Fig. 3 shows one of the simulated environments, and the real one.
- Although the position of the base cell within the environment is not given to the robot, the initial pose is not estimated globally either. The robot is always placed in the base cell in the same way as in the simulated world, and it takes its own initial heading as the reference direction of the map it builds, so that the map frame is defined relative to the starting pose. Students therefore do not have to solve a global initial-localisation problem: they only have to track position and orientation by odometry, which should be identical in simulation and on the real robot.
- Localization must be carried out in two ways. First, using the Webots global localisation through the "supervisor" feature, to simplify development and debugging. The final implementation must replace this with local odometry based on the robot's wheel encoders and gyroscope, so that the same solution can later be transferred to the physical robot.
- During the map creation phase, the robot builds a binary occupancy grid, distinguishing between free and occupied cells. LiDAR readings are taken with the robot stopped at the centre of each cell, allowing the system to detect walls or obstacles in adjacent cells and update the internal map.
- Once the map has been created, the robot must use it to compute an optimal path from its current position to a user-defined destination cell. This second stage assumes a static environment, so the occupancy grid generated during exploration remains valid for navigation.
- Before using the real robot, students test the complete behaviour in Webots: discrete motion, map updating, localisation, obstacle detection and path following. The goal is to ensure that the algorithm works reliably in the simulated environment shown in the accompanying figure.
- After validation in simulation, the students transfer their developed code to the TurtleBot3. This step requires adapting the implementation so that it uses the real robot interfaces instead of the simulated Webots devices.

At the end of the semester, students were asked to fill in a survey to get their feedback regarding library usage. Table 1 contains qualitative opinions with regards to the advantages of using this library as compared to ROS. It can be observed that most of the students confirm the simplicity of this approach as the main advantage.

Although it was not the main objective here, Fig.4 shows an interesting result we have obtained from this experience. Even in the case that these students did not have a deep informatics background, they highlight the use of the real robot as compared to only the simulated one. We want to point out this, because many new courses at this level tend to avoid physical robots due to its complexity, but students' opinion is really positive towards its use.

4 Conclusions

This paper presented a custom Python library that enables the direct transfer of Webots controllers to a physical TurtleBot3 Burger without relying on

Based on your knowledge and experience about ROS, what main advantages did you find in programming and running the TurtleBot3 directly in pure Python?
- Simpler and more focused on the control logic.
- When programming directly in Python, you do not need to think about the distribution of nodes or communication mechanisms, so the programming style is closer to the type of programming we use in the rest of the subjects.
- I do not have to think about packages, services, messages, publishing, subscription, etc.
- The main advantage I found in Python is the ease of integrating code libraries, such as NumPy or priority queues.
- The simplicity of using only a Python script and not having to deal with ROS packages.

Table 1. Students' responses on the advantages of programming and running the TurtleBot3 directly in Python. It must pointed out that this group of students has basic ROS knowledge due to a theoretical class they receive in other subject of the master.

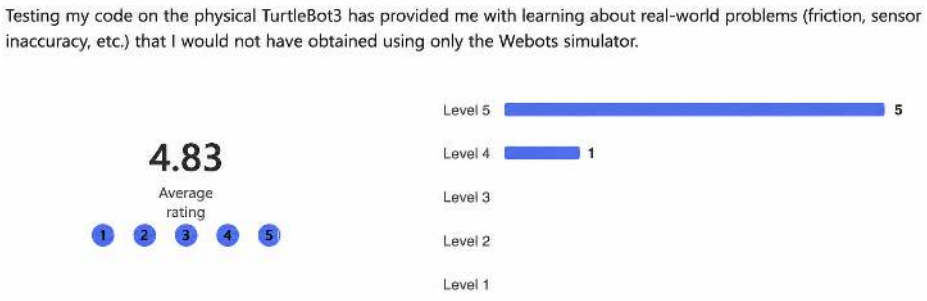


Fig. 4. Student's answer related with the use of the real robot.

ROS. The objective is not to replace ROS as a standard robotics middleware, but to provide a simpler educational pathway for courses where the main focus is autonomous robotics rather than middleware configuration. The library was applied in a master-level course on "Introductory Autonomous Vehicles", where students addressed a complete task involving map creation, path planning and navigation in both simulated and real environments. Student feedback suggests that the approach reduces the perceived complexity associated with ROS packages, messages, publishers, subscribers and distributed nodes, allowing learners to focus more directly on control and navigation logic.

Regarding scalability, the library currently covers the devices required by the course: wheel motors and encoders, IMU and LiDAR. The object-oriented design encapsulates each device in its own class, so adding a new one connected through the serial or USB interfaces only requires a class mirroring the corresponding Webots node. Higher-bandwidth sensors such as the RGB-D cameras of other

TurtleBot3 variants would need a different acquisition path and are left as future work.

Acknowledgments. This work was supported by the Spanish Ministry of Science, Innovation and Universities through the grant FPI-UDC2025/06 funded by PID2024-162223OB-I00, and CITIC (Xunta de Galicia - Convenio para o desenvolvemento de accións estratéxicas de I+D+i 2025-2026).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software. vol. 3, p. 5 (2009)
2. Ryalat, M., Almtireen, N., Al-refai, G., Elmoaqet, H., Rawashdeh, N.: Research and Education in Robotics: A Comprehensive Review, Trends, Challenges, and Future Directions. Journal of Sensor and Actuator Networks, 14, 76. (2025) https://doi.org/10.3390/jsan14040076
3. Mamatnabiyev, Z., Chronis, C., Varlamis, I., Himeur, Y., Zhaparov, M.: A Holistic Approach to Use Educational Robots for Supporting Computer Science Courses. Computers, 13, 102 (2024) https://doi.org/10.3390/computers13040102
4. Crompton, H., Burke, D. Artificial intelligence in higher education: the state of the field. Int J Educ Technol High Educ 20, 22 (2023). https://doi.org/10.1186/s41239-023-00392-8
5. Roldán-Álvarez, D., Cañas, J. M., Valladares, D., Arias-Perez, P., Mahna, S.: Unibotics: open ROS-based online framework for practical learning of robotics in higher education. Multimedia Tools and Applications, 83(17), 52841-52866 (2024)
6. Chen, L. et al. (2025). Teaching Service Robotics with ROS and Unibotics Web Framework in Higher Education. In: Balogh, R., Obdržálek, D., Fachantidis, N. (eds) Robotics in Education. RiE 2025. Lecture Notes in Networks and Systems, vol 1544. Springer, Cham. https://doi.org/10.1007/978-3-031-98762-5_43
7. University of Coruña, Industrial Informatics and Robotics Master, accessed 10/06/2025, https://estudos.udc.es/en/study/detail/4556V01
8. Cyberbotics Ltd.: Webots: open-source robot simulator. https://cyberbotics.com, last accessed 10/06/2025
9. Tselegkaridis, S., Sapounidis, T.: Simulators in Educational Robotics: A Review. Education Sciences 11, 11 (2021) https://doi.org/10.3390/educsci11010011
10. Farley, A., Wang, J., Marshall, J. A.: How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion. Simulation Modelling Practice and Theory, 120, (2022). 102629.
11. ROBOTIS: Dynamixel2Arduino: DYNAMIXEL protocol library for Arduino. https://github.com/ROBOTIS-GIT/Dynamixel2Arduino, last accessed 10/06/2025
12. Cyberbotics Ltd.: Webots Reference Manual — API. https://cyberbotics.com/doc/reference/nodes-and-api-functions, last accessed 10/06/2025

Explainable AI for medical imaging

Jose Luis Garcia Salas[1][0000−0002−8584−4844] and Pilar Bachiller Burgos[2][0000−0001−8699−7749]

Tecnología de los Computadores y de las Comunicaciones, Laboratorio de robótica y visión artificial, Escuela Politécnica, Universidad de Extremadura, Avenida de la Universidad, S/N, Cáceres, 10003, Spain joselgs96@unex.es, pilarb@unex.es

Abstract. In recent years, deep learning has emerged as a pivotal tool for assisting in disciplines and domains where the data involved is very sensitive and difficult to process. In the field of medical imaging and computer-aided diagnosis (CAD), the advances achieved by modern architectures regarding disease classification, lesion segmentation, and related healthcare tasks have established AI as a cornerstone technology to support medical experts in improving the clinical workflow in their daily routine.

Despite the promising results given by these tools, due to the highly complex nature of the diverse types of medical images, ensuring reliability becomes essential for the expert to provide a confident diagnosis. The "**black-box**" nature of the deep learning models makes them, in many situations, very hard to interpret for medical clinicians, causing them to doubt the model results and decreasing their confidence in AI tools.

Explainable artificial intelligence (XAI) has recently gained significant importance as an alternative to address this situation. XAI provides methods to make the reasoning process of deep learning models interpretable and comprehensible for non-AI experts.

This thesis proposal delves into the XAI paradigm applied to CAD, prioritizing breast lesion detection, going through post-hoc evaluation methods for interpreting model results, and the design of interpretable models for transparency, focusing on their limitations and exploring new research avenues to foster the clinical acceptance and seamless integration of XAI into future CAD systems.

Keywords: XAI · Post hoc methods · Intrinsic explainability.

1 Deep neural networks in medical image analysis

The evolution of CAD systems increased exponentially in the last decade, leveraging the potential of deep learning models for lesion diagnosis, which has significantly facilitated the optimization of screening workflows. McKinney et al. [14] demonstrated promising results in breast cancer detection, achieving near medical-expert-level; however, convolutional approaches have limitations when modelling long-range spatial dependencies and highly variable anatomical structures in medical images.

To address this challenge, many recent works have included attention-based techniques to improve the capture of spatial and contextual information [15]. The inclusion of these mechanisms and its excellent performance motivated the evolution of this type of architectures to new models such as Vision Transformers(ViTs)[6] and its integration with convolutional approaches like UNETR[7].

This progress promoted clinical evaluations to confirm that incorporating these tools into the CAD workflow and population programs can improve the early diagnosis and reduce the medical expert workload [11].

Despite these advances and achieved clinical benefits, most deep learning models suffer from opacity due to their black-box nature and increasing complexity. This challenge has limited the adoption of these models in clinical practice. Understanding the prediction has become increasingly important since the advent of deep learning in CAD systems where model predictions must be interpretable and transparent.

Explainability provides information about the factors that influence the model results, supporting clinical decision-making, helping medical experts understand these models, and increasing trust in AI systems.

2 Evolution of XAI in medical imaging

The Explainable Artificial Intelligence (XAI) paradigm gathers different techniques to provide transparent explanations in sensitive application fields, such as medical imaging, uncovering the reasoning behind the models' decisions.

In computer vision tasks, the XAI techniques are based on giving spatial explanations, identifying the most important regions, and justifying the model's predictions.

XAI methods can be classified into two main categories based on whether explanations are generated: **Extrinsic** or **Post-hoc** methods, and **Intrinsic** methods.

2.1 Post-hoc methods

Post-hoc methods rely on generating explanations through analysis applied to an already trained model to complement the model's results. Some of the most representative methods are SHAP [13] and LIME[17], which are perturbation-based methods checking the influence of input features by analysing the model outputs. Another notable type is the gradient-based approach, such as Integrated Gradients [20]. Related to image processing, Grad-CAM [19] has been a cornerstone, deriving explanations from the model gradients and activations.

Even though these tools have been adopted due to their flexibility and compatibility with existing architectures, the explanations and reasoning rely on the final predictions of the model, being independent of the decision process and not always representing the model's real reasoning [1][8].

These challenges have promoted the development of architectures intrinsically interpretable by design, making the reasoning for decision-making more transparent and understandable [18].

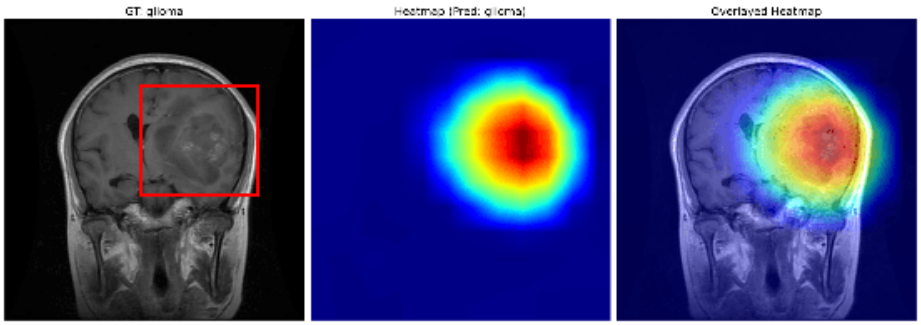


Fig. 1. Example of Grad-CAM application. Source: https://learnopencv.com/intro-to-gradcam/

2.2 Intrinsic explainability

The post-hoc methods explanations are external interpretations of the results given by the model [18]. This dependency can make the model's explanations hard to understand or even not trustworthy in sensitive data domains like medical imaging: If the model does not generalize well or make assumptions regarding the data, this can lead to a false positive or false negative diagnosis, which is a catastrophic situation. Intrinsic explainability is based on the premise that the reasoning process of the model can be understood from its own structure [5]. Intrinsic explainability emerges as an important alternative to design architectures to be more transparent and clear, allowing the experts to understand which parts of the data the network is focusing on, making the diagnosis more reliable and accurate.

3 Towards Transparent Medical Image Diagnosis: Spatial explainability in XAI

In recent times, new lines of investigation have emerged for intrinsic XAI applied to medical imaging:

- One of the principal approaches is **prototype-based architectures** [5]. These models provide explanations through the identification of image regions compared to diagnostically relevant prototypes. The performance of these architectures depends on the representation capacity of the learned prototypes [16].
- Another promising paradigm is presented in **Concept Bottleneck Models** [10], which are designed to learn understandable concept representations for humans. Some of them use concept-specific prototypes to locate concepts within the images [9], others use spatially-aware CBM to project backbone features onto concept maps to provide better concept explanations [3].

- The integration of attention mechanisms into the decision process has been another line of investigation [12] [21]. Other works have included MIL-based approaches to generate maps to represent the importance of different regions of the image [2].
- Although these methods provide promising results, it has been noted in recent years that attention represents only a part of the decision process. Due to this, it is important to ensure that they really correspond to the most significant region for the final prediction. One important work that explores these distinctions between attention and real contribution is **AttriMIL** [4], highlighting the importance of individual instances on the model's output.

4 Conclusions

Even though explainable artificial intelligence represents a crucial line of research in contemporary medical imaging, significant room for improvement remains regarding intrinsic designs that render the model's decision-making process transparent. In this clinical context, it is essential to simultaneously evaluate both predictive performance and the quality of explanations, as optimizing either metric in isolation yields a suboptimal trade-off that is insufficient for real-world medical deployment and clinical diagnosis.

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity Checks for Saliency Maps. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 9525–9536. Curran Associates Inc., Red Hook, NY, USA (2018)
2. Barbosa, D., Ferreira, M., Junior, G.B., Salgado, M., Cunha, A.: Multiple instance learning in medical images: A systematic review. IEEE Access **12**, 78409–78422 (2024)
3. Benou, I., Raviv, T.R.: Show and tell: Visually explainable deep neural nets via spatially-aware concept bottleneck models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 30063–30072 (2025)
4. Cai, L., Huang, S., Zhang, Y., Lu, J., Zhang, Y.: AttriMIL: Revisiting Attention-Based Multiple Instance Learning for Whole-Slide Pathological Image Classification From a Perspective of Instance Attributes. Medical Image Analysis **103**, 103631 (2025)
5. Chen, C., Li, O., Tao, C., Barnett, A.J., Su, J., Rudin, C.: This Looks Like That: Deep Learning for Interpretable Image Recognition. In: Advances in Neural Information Processing Systems. vol. 32, pp. 8930–8941. Curran Associates, Inc. (2019)
6. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: International Conference on Learning Representations (ICLR) (2021)
7. Hatamizadeh, A., Tang, Y., Nath, V., Yang, D., Myronenko, A., Landman, B., Roth, H.R., Xu, D.: UNETR: Transformers for 3D Medical Image Segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 574–584 (2022)

8. Huang, X., Marques-Silva, J.: On the failings of Shapley values for explainability. International Journal of Approximate Reasoning **171**, 109112 (2024)
9. Jeon, S., Lee, H., Kim, E., Lee, S., Zhang, B.T., Hwang, I.: Locality-aware concept bottleneck model. arXiv preprint arXiv:2508.14562 (2025)
10. Koh, P.W., Nguyen, T., Tang, Y.S., Mussmann, S., Pierson, E., Kim, B., Liang, P.: Concept Bottleneck Models. In: Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 5338–5348. PMLR (2020)
11. Lång, K., Josefsson, V., Larsson, A.M., Larsson, S., Högberg, C., Sartor, H., Hofvind, S., Andersson, I., Rosso, A.: Artificial intelligence-supported screen reading versus standard double reading in the Mammography Screening with Artificial Intelligence trial (MASAI): a clinical safety analysis of a randomised, controlled, non-inferiority, single-blinded, screening accuracy study
12. Leem, S., Seo, H.: Attention guided CAM: visual explanations of vision transformer guided by self-attention. In: Proceedings of the AAAI conference on artificial intelligence. vol. 38, pp. 2956–2964 (2024)
13. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 4768–4777. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
14. McKinney, S.M., Sieniek, M., Godbole, V., Godwin, J., Antropova, N., Ashrafian, H., Back, T., Chesus, M., Corrado, G.S., Darzi, A., et al.: International evaluation of an AI system for breast cancer screening. Nature **577**(7788), 89–94 (2020)
15. Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., et al.: Attention U-Net: Learning Where to Look for the Pancreas. arXiv preprint arXiv:1804.03999 (2018)
16. Patrício, C., Neves, J.C., Teixeira, L.F.: Explainable deep learning methods in medical image classification: A survey. ACM Computing Surveys **56**(4), 1–41 (2023)
17. Ribeiro, M.T., Singh, S., Guestrin, C.: Why Should I Trust You?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144. KDD '16, Association for Computing Machinery, New York, NY, USA (2016)
18. Rudin, C.: Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
19. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 618–626 (2017)
20. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic Attribution for Deep Networks. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 3319–3328. PMLR (2017)
21. Torres, F., Zhang, H., Sicre, R., Ayache, S., Avrithis, Y.: CA-Stream: Attention-Based Pooling for Interpretable Image Recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 8206–8211 (2024)

Predictive navigation for agricultural robots

Víctor Romero Bautista[1[0009−0008−7622−5755]], Luis V. Calderita Estévez[2,3[0000−0003−0784−4102]], Pablo Bustos García De Castro[2,3[0000−0003−3202−2493]], and Pedro Núñes Trujillo[3[0000−0002−3615−8833]]

RoboLab - Robotics and Artificial Vision Laboratory, University of Extremadura, Avenida de la Universidad S/N, Cáceres, Spain
{victor.romerobaut, lvcalderita, pbustos, pnuntru}@unex.es

Abstract. Autonomous robot navigation systems in unstructured agricultural environments remains an open problem. Repetitive patterns, dense canopy, illumination changes, and uneven terrain cause failures in the navigation algorithms. This thesis proposes a two-stage framework in which a robot first builds a map of its environment to localize itself and then learns an internal model of the world that predicts how the environment changes as it moves. These two stages will be internally linked through active inference as a planning layer, which will allow the robot to reason about future states and their degree of confidence, and act to reduce uncertainty when necessary. The central contribution is the epistemic-pragmatic decomposition of Expected Free Energy over agricultural world model rollouts, which enables the robot to detect when its predictions are unreliable and act to resolve that uncertainty before committing to a trajectory.

Keywords: SLAM · World models · Active inference.

1 Introduction

Autonomous navigation in unstructured agricultural environments arises challenges that current robotic systems have not fully resolved. Uneven terrain, non-rigid vegetation, seasonal appearance changes, and visual aliasing, create conditions of persistent uncertainty that undermine standard localization methods [8]. GNSS-based localization systems degrades severely under dense canopy and orchard structures. SLAM strategies offers a sensor independent alternative, but suffers from perceptual aliasing and generates drift over time, sensor fusion can attenuate but not eliminate this inconvenience 4, 9.

Beyond localization, autonomous navigation requires the capacity to anticipate the consequences of actions before executing them. Reactive systems that respond only to current observations cannot plan around occlusions or reason about unseen regions. Here, an internal world model can be employed to image possible future scenarios conditioned on actions, and choice the action that best balances the goal pursuit with the reduction of uncertainty.

This thesis proposes the combination of SLAM for map construction and a latent world model which learn the dynamics from agricultural egocentric video

data, enabling predictive trajectory planning in latent space. These components conform a predictive navigation architecture designed to operate under agricultural conditions.

2 Thesis proposal

2.1 Theoretical framework

SLAM Simultaneous Localization And Mapping (SLAM), is a technique that allows a robot to estimate a map of an unknown environment simultaneously as it locates itself on the map [3]. State-of-the-art SLAM methods are commonly comprised by front-end, back-end, mapping, and loop closure modules. The front-end similarly to odometry, addresses the problem of tracking the trajectory, which usually consist on movement estimation. The back-end module optimizes globally the information constructed by the front-end. Loop closure attempts to eliminate the drift usually by detecting the current scene into historical frames to add constraints and eliminate accumulated errors. In mapping, the global map of the environment is built [2].

World models The concept of a world model coined by [6], involves abstracting the external world perceived to gain a deep understanding of the mechanisms, learning the dynamics of the world [5], can envision possible future states to inform decision-making as in 1. A world model is typically composed by an encoder which maps observations s_t and actions a_t to latent states z_t, and a latent world model predicts future conditioned actions $p\left(z_{t+1}|z_t, a_t\right)$.

Active inference Consists of a generative model $P\left(\tilde{o}, \tilde{s}, \tilde{a}\right)$, where (s) are the internal belief states, (a) represents the perceived actions, and (o) corresponds to the observations of the world. The free energy is defined as follows.

$$F = \mathbb{E}_Q\left[logQ\left(\tilde{s}\right) - logP\left(\tilde{o}, \tilde{s}, \tilde{a}\right)\right] = D_{KL}\left(Q\left(\tilde{s}||P\left(\tilde{s}, \tilde{a}\right)\right)\right) - \mathbb{E}_Q\left[logP\left(\tilde{o}|\tilde{s}\right)\right], \quad (1)$$

where Q is the approximation to posterior distribution; the second equality represents the free energy as the negative evidence lower bound (ELBO) which also appears in the VAE [7]. Thus, the action of an active inference agent can be represented as

$$P\left(\pi\right) = \sigma\left(-\gamma G\left(\pi\right)\right), \quad G\left(\pi\right) = \sum_{\tau=t}^{t+H} G\left(\pi|\tau\right), \quad (2)$$

here, π corresponds to a policy which is a sequence of actions, starting in time t_0 up to horizon time H. σ means a softmax function with precision γ.

Deep active inference Consists of allowing the agent to learn by itself what the proper configuration of its belief space should be. To do this, deep neural networks are employed to generate a variety of probability distributions, which are trained using the free energy. Thus, the posterior variational distribution

for a time step $Q\left(s_t|s_{t-1}, a_{t-1}, o_t\right)$ is approximated with an Artificial Neural Network (ANN) $q_\phi\left(s_t|s_{t-1}, a_{t-1}, o_t\right)$. The likelihood $P\left(o_t|s_t\right)$ is approximated as $p_\xi(o_t|s_t)$. The prior $P\left(s_t|s_{t-1}, a_{t-1}\right)$ is approximated by $p_\theta\left(s_t|s_{t-1}, a_{t-1}\right)$. These ANNs are trained in an end-to-end fashion by using the free energy as the loss function, which is defined as

$$\mathcal{F}_t = -logp_\xi\left(o_t|s_t\right) + D_{KL}[p_\phi(s_t|s_{t-1}, a_{t-1}, o_t) \parallel p_\theta\left(s_t|s_{t-1}, a_{t-1}\right)] \tag{3}$$

Trained models can be used to perform active inference and determine what next action the agent should take. This can be done by generating trajectories for different policies using p_θ and p_ξ, computing the expected free energy G, and selecting the policy action that achieves the lowest G.

2.2 Hypothesis

A robot equipped with a generative world model and a metric map constructed by SLAM, and which plans by minimizing the expected free energy, can navigate autonomously in unstructured agricultural environments anticipating possible future scenarios.

2.3 Objectives

General objective

- To develop a predictive autonomous navigation architecture for agricultural robots that integrates SLAM for map construction and localization, a latent world model for imagining future trajectories before execution, and active inference as the planning layer. Evaluate its performance under agricultural conditions.

Specific objectives

- Develop at SLAM pipeline for a robot that maintains reliable localization and map under agricultural conditions.
- Design and train a latent world model on agricultural egocentric video that predicts future observations from the map as the substrate for predictive planning.
- Evaluate a compare the integrated architecture with state-of-the-art methods.

3 Conclusion

This thesis proposes a predictive autonomous navigation architecture for agricultural robots, structured in two complementary layers: SLAM as a geometric

foundation for localization and map construction, and a world model as a planning engine that simulates future trajectories in latent space before executing them. The integration of both under the principle of free energy will allow the robot to anticipate the consequences of its actions, quantify its own uncertainty, and act to reduce it when necessary. The resulting method is expected to be capable of navigating robustly in a highly uncertain agricultural environment.

Acknowledgments. This work is supported by the AIGreenBots project (II0446), founded by the European Union's Horizon Europe research and innovation program under the Marie Sklodowska-Curie Actions (Doctoral Networks) grant agreement.

References

1. Bar, A., Zhou, G., Tran, D., Darrell, T., LeCun, Y.: Navigation world models (2024), https://arxiv.org/abs/2412.03572
2. Bresson, G., Alsayed, Z., Yu, L., Glaser, S.: Simultaneous localization and mapping: A survey of current trends in autonomous driving. IEEE Transactions on Intelligent Vehicles **2**(3), 194–220 (2017). https://doi.org/10.1109/TIV.2017.2749181
3. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics **32**(6), 1309–1332 (2016). https://doi.org/10.1109/TRO.2016.2624754
4. Cremona, J., Comelli, R., Pire, T.: Experimental evaluation of visual-inertial odometry systems for arable farming. Journal of Field Robotics **39**(7), 1121–1135 (2022). https://doi.org/https://doi.org/10.1002/rob.22099, https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22099
5. Ding, J., Zhang, Y., Shang, Y., Zhang, Y., Zong, Z., Feng, J., Yuan, Y., Su, H., Li, N., Sukiennik, N., Xu, F., Li, Y.: Understanding world or predicting future? a comprehensive survey of world models. ACM Comput. Surv. **58**(3) (Sep 2025). https://doi.org/10.1145/3746449, https://doi.org/10.1145/3746449
6. Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf
7. Mazzaglia, P., Verbelen, T., Catal, O., Dhoedt, B.: The free energy principle for perception and action: A deep learning perspective. Entropy **24**(2), 301 (2022)
8. Moshayedi, A.J., Sohail Khan, A., Yang, Y., Hu, J., Kolahdooz, A.: Robots in agriculture: Revolutionizing farming practices. EAI Endorsed Transactions on AI and Robotics **3** (Jun 2024). https://doi.org/10.4108/airo.5855, https://publications.eai.eu/index.php/airo/article/view/5855
9. Xu, S., Zhong, Z., Barros, T., Coombes, M., Premebida, C., Wu, H., Liu, C.: Hortimulti: A multi-sensor dataset for localisation and mapping in horticultural polytunnels (2026), https://arxiv.org/abs/2603.20150

The Use of Robotics in Occupational Therapy for Cognitive Stimulation and the Prevention of Cognitive Decline: A Person-Centred Approach for Older Adults

Nerea Aparicio-Diacosta[1] [0009-0009-7702-0089] Alicia Condón-Pérez[1] [0009-0007-3800-4831] Zoraida Clavijo-Chamorro[1] [0000-0002-8603-8370] and Trinidad Rodríguez-Domínguez[1] [0000-0001-7594-0011]

[1] Nursing and Occupational Therapy College, Extremadura University, Cáceres, Spain.

trdomin@unex.es

Abstract. The ageing population and the increasing prevalence of cognitive impairment have driven the search for innovative interventions aimed at preserving the independence and quality of life of older people. In this context, robotics has emerged as a tool with growing potential within Occupational Therapy. The purpose of this work is to describe, using a narrative based on scientific evidence, the occupational therapy process for the use of person-centered robotics. Considering the socio-emotional aspects, the applications of assistance and cognitive stimulation of robots in older people with cognitive impairment and the characteristics of their occupational performance, the guidelines for intervention in occupational therapy in older people facing cognitive impairment are presented. Robotics is a complementary resource that can promote cognitive stimulation, occupational participation, and functional autonomy, provided its use is integrated into meaningful occupations and within the clinical reasoning of occupational therapy, and is carried out under a person-centered robotics approach.

Keywords: Occupational Therapy, Robotics, Dementia, Cognitive Impairment, Activities of Daily Living, Older adults, Cognitive stimulation, Efficacy.

1 Introduction

Please Ageing may be accompanied by a progressive decline in cognitive functions such as memory, attention or language, although this does not necessarily imply the presence of dementia. However, as this decline progresses, it can compromise independence and increase dependency, which poses a growing challenge in the context of an ageing population. [1]

Distinguishing between normal ageing and dementia is essential, as the latter involves a significant impairment of independence and key cognitive functions such as orientation and reasoning. Early detection enables appropriate intervention and helps to preserve the highest possible level of functionality. [2]

In this context, Occupational Therapy and robotics, using a person-centred approach, can help maintain participation in Activities of Daily Living (ADL). An occupational profile assessment is used to determine the use of assistive technologies tailored to the user's needs, always as a complement to professional intervention. [3]

2 Methodology

A search for scientific evidence has been conducted, and an occupational therapy process has been developed to guide the application of person-centered robotics. This process considers socio-emotional aspects, assistive applications, cognitive stimulation, and the characteristics of occupational performance, all with the aim of demonstrating how person-centered robotics technology should be applied to older adults with cognitive impairment.
Studies were included that related to the use of robots for emotional support, assistance with Activities of Daily Living (ADL) and cognitive stimulation in older people with cognitive impairment. Based on the information obtained in this search, this paper presents a person-centred Occupational Therapy intervention model using robots, aimed at older people with mild to moderate cognitive impairment. It also examines the potential benefits that its implementation could bring to the therapeutic management of this population.

3 Development

Depending on their therapeutic objectives, robots can be classified into three main groups.
● Socio-emotional robots, designed to encourage social interaction and reduce feelings of loneliness. Among the most representative examples are PARO, AIBO, Babyloid and EBO. [4]
● Assistive robots, designed to facilitate Activities of Daily Living (ADL) and promote functional independence. Notable examples in this group include PaPeRo, GrowMu, Héctor and Giraff. [5]
● Cognitive stimulation robots, which aim to enhance functions such as memory, attention and language through interactive activities. These include Pepper, Mini and Eldertoy. [5]
The evidence reviewed suggests that the choice of robot should not be based solely on its technological characteristics but should be integrated into the clinical reasoning process inherent to Occupational Therapy. [6]
A five-stage model is proposed: [6]
1. Assessment of cognitive status, occupational performance and social participation.
2. Identification of therapeutic needs and objectives.
3. Selection of the most appropriate type of robot.
4. Implementation of the intervention.

5. Monitoring and ongoing adaptation.
These results highlight that the success of the intervention depends primarily on the ability to adapt the technology to the user's individual characteristics and needs. [6]
Occupational Therapy aims to encourage people to participate in activities that give meaning to their lives. According to the American Occupational Therapy Association's (AOTA) 'Framework for Occupational Therapy Practice: Domain and Process', intervention should be geared towards promoting participation in meaningful occupations, taking into account each individual's needs, abilities and context. In this regard, the integration of technology must be tailored to the user's life history, preferences and personal goals, whilst respecting their identity and promoting their functionality. [3]
Under this person-centred approach, the user is placed at the heart of the therapeutic process and participates in decision-making regarding their treatment. In this way, the clinical assessment is complemented by those aspects that the individual themselves considers relevant to their daily life, promoting holistic and personalized care. [7]
The integration of robotics into these interventions must address people's real needs and help to improve their functional ability, independence and quality of life. Furthermore, its incorporation must be carried out in a manner consistent with the principles of Occupational Therapy and taking into account the ethical implications arising from its use, so that technological innovation is always at the service of the individual's dignity and well-being. [8]

4 Discassion

The studies reviewed indicate that robots are a promising complementary tool in Occupational Therapy interventions. [9] Robots such as PARO and Pepper have shown particularly favorable results in terms of emotional well-being, social interaction and motivation. [5] Furthermore, assistive robots help to maintain independence and reduce the burden on carers. [9]
However, most studies have small sample sizes and limited intervention periods, making it difficult to draw conclusions about long-term effects. Furthermore, factors such as cost, accessibility and the need for specific training for professionals remain significant barriers to their implementation. [10]
The results suggest that technological complexity does not guarantee better therapeutic outcomes. On the contrary, effectiveness appears to depend on personalization and the integration of robotics into activities that are meaningful to the individual. [11]

5 Conclusion

Robots represent a complementary resource with great potential for promoting cognitive stimulation, emotional wellbeing and independence among older people with cognitive impairment.

The success of these interventions depends more on their adaptation to individual needs than on the technological sophistication of the devices. In this regard, a selection model is proposed based on a person-centred approach and the user's occupational goals. Studies with larger samples and long-term follow-ups are needed to consolidate the available evidence and promote the safe and effective integration of robotics into Occupational Therapy practice.

Acknowledgments. This work was funded by Project 0124 EUROAGE MAS 4 E (POCTEP 2021-27, ERDF funds).

Disclosure of Interests. The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Benavides-Caro CA. Deterioro cognitivo en el adulto mayor. Rev Mex Anestesiol. 27 de junio de 2017;40(2):107-12.
2. Labos E, Cristalli D, Deschle F, Colli L, Berrios W, Dorman G, et al. [Not Available]. Vertex. 10 de enero de 2026;36(170, oct.dic.):41-63. doi:10.53680/vertex.v36i170.943 PubMed PMID: 41528085.
3. Marco de Trabajo. Terapia Ocupacional (4° Ed. 2020) [Internet]. Disponible en: https://www.academia.edu/88977775/Marco_de_Trabajo_Terapia_Ocupacional_4_Ed_2020
4. Dragomir O, Iglesias Gozalo MJ. Nuevas intervenciones de terapia ocupacional en demencias. Una revisión sobre el uso de las tecnologías en Asia Oriental. Zaragoza: Universidad de Zaragoza; 2016.
5. Salichs San José E. Ayuda a personas mayores con un robot social: estimulación cognitiva [Internet]. octubre de 2019. Disponible en: https://hdl.handle.net/10016/31375.
6. Bas Berná E. Plan Centrado en la persona: el papel fundamental de la Terapia Ocupacional [Internet]. 9 de septiembre de 2016. Disponible en: http://dspace.umh.es/handle/11000/3614.
7. Torres, L. (2022, enero 31). Guía de tecnología centrada en la persona. Plena Inclusión. https://riberdis.cedid.es/items/f1598b08-3f3a-4438-95fe-f6ad16cd75b0
8. Nuevas tecnologías en terapia ocupacional: revisión bibliográfica - Universidad de Zaragoza Repository. (2014). Universidad de Zaragoza. https://zaguan.unizar.es/record/13931
9. Boada, J. P., Maestre, B. R., & Genís, C. T. (2021). The ethical issues of social assistive robotics: A critical literature review. Technology in Society, 67(101726), 101726. https://doi.org/10.1016/j.techsoc.2021.101726
10. Onrubia Martínez V, Roig Vila R, Cazorla Quevedo M. El papel de la robótica social en la sociedad: avances y desafíos. En: Humanidades y conocimiento como ejes del pensamiento crítico, 2025, ISBN 978-84-1079-133-6, págs. 118-126 [Internet]. Octaedro; 2025. p. 118-26. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=10198448
11. Vista de Desafíos y avances en la robótica de rehabilitación: un enfoque en la inclusión y la innovación. (s/f). Edu.ec. Recuperado el 21 de julio de 2026, de https://institutojubones.edu.ec/ojs/index.php/societec/es/article/view/558/891

A Digital Shadow Architecture for Real-Time Spatial Synchronization

Sergio Suescun-Ferrandiz*[0009−0008−1521−0031], Carlos Zambrana-Navajas[0009−0008−0697−4822], Francisco Gomez-Donoso[0000−0002−7830−2661], and Miguel Cazorla[0000−0001−6805−3633]

University Institute for Computer Research. University of Alicante.
Ctra. San Vicente del Raspeig SN, 03690, Alicante. Spain.
{sergio.suescun, carlos.zambrana, fgomez, miguel.cazorla}@ua.es
*Corresponding Author

Abstract. This paper presents a lightweight Digital Shadow approach for maintaining spatial consistency between a physical robotic workspace and its virtual representation. Instead of implementing a bidirectional Digital Twin, the proposed system focuses on safe and decoupled real-to-virtual synchronization, where the virtual scene is updated from observations of the physical robot without affecting its execution.
The system uses an overhead vision setup with fiducial markers to localize a mobile robot inside a calibrated arena. After an initial camera-to-arena calibration, the robot pose is continuously estimated, constrained to planar motion, and transmitted through a ROS 2-based pipeline to update the virtual model. The architecture is evaluated in terms of tracking accuracy, pose stability, and synchronization latency. Experimental results show low measurement variability and an average end-to-end delay below 400 ms, indicating that the proposed approach is suitable for real-time monitoring of physical-to-virtual consistency in robotic workspaces.

Keywords: Digital Shadow · Spatial Synchronization · Robotic Workspaces · Vision-Based Tracking · Fiducial Markers

1 Introduction

Virtual representations are useful tools for supervising robotic experiments, analyzing system behavior, and comparing real executions with simulated environments. For these representations to be meaningful, the spatial state of the physical robot must remain aligned with its virtual counterpart. This is challenging in real workspaces, where wheel slippage, surface irregularities, mechanical tolerances, and odometry drift can cause the estimated robot position to diverge from its actual motion.

In many experimental scenarios, the goal is not to control the physical robot from simulation, but to obtain a reliable virtual reflection of what is happening in the real environment. A Digital Shadow is well suited to this purpose, since it

maintains a unidirectional flow of information from the physical system to the virtual one. This avoids closing the control loop through the simulator and keeps the monitoring architecture independent from the robot controller.

The main challenge addressed in this work is the construction of a simple and robust spatial synchronization pipeline. To this end, the proposed system relies on external visual perception rather than internal odometry. A set of fixed fiducial markers is used to estimate the pose of the overhead camera with respect to the arena, while a marker mounted on the robot provides its pose during operation. This pose is then expressed in the workspace reference frame and used to update the virtual representation in real time.

The contribution of this paper is a low-cost and decoupled Digital Shadow architecture for robotic workspaces, together with an experimental evaluation of its synchronization performance. The system is assessed through spatial tracking accuracy, measurement stability, and end-to-end latency, providing insight into the practical viability of external vision as a basis for real-time physical-to-virtual monitoring.

2 Related Work

The concept of the Digital Twin (DT) has been extensively explored in robotics for monitoring, predictive maintenance, and simulation. Following the taxonomy established by Kritzinger et al., a strict DT requires bidirectional, automated data flow between the physical and virtual entities [4]. However, many current implementations in robotics function primarily as Digital Shadows (DS), where data flows unidirectionally from the physical workspace to update the virtual model [4]. While comprehensive DTs attempt to simulate full kinematic and dynamic responses, this level of fidelity imposes heavy computational burdens [3]. As noted by Jones et al., an exhaustive high-fidelity implementation where parameters for every aspect of the physical twin are captured introduces severe challenges in elements of the system such as network speeds and computational processing power [3]. Consequently, for applications focused purely on tracking the robot's general position within a workspace, a lightweight Digital Shadow that does not rely on the robot's internal controllers offers a more computationally efficient approach.

For workspace-level localization, external tracking systems provide absolute pose estimates independently of the robot's onboard sensors. While optical Motion Capture (MoCap) systems offer the highest accuracy, they require dedicated infrastructure and high deployment costs [6]. In contrast, overhead monocular cameras combined with fiducial markers such as ArUco and AprilTag enable robust, low-cost pose estimation and have become widely adopted for robot localization and camera pose estimation in structured indoor environments [2,7,9,8]. This makes marker-based overhead vision particularly suitable for Digital Shadow systems, where the objective is to maintain spatial consistency between physical and virtual workspaces rather than achieve MoCap-level precision.

Addressing this integration requires robust distributed simulation architectures. The robotics community relies heavily on the Robot Operating System 2 (ROS 2) for distributed communication [5]. Concurrently, NVIDIA Isaac Sim has emerged as a leading platform for physically accurate robotic simulation [1]. While ROS 2 enables the integration of perception, control, and visualization modules through a distributed middleware, tightly coupling the robot's internal state with the simulation can increase computational dependencies. Motivated by these considerations, our Digital Shadow adopts a modular architecture in which external vision, communication, and visualization are executed as independent ROS 2 nodes. This design keeps localization independent of the robot's internal control pipeline while supporting real-time synchronization between the physical and virtual workspaces.

3 Methodology

This work proposes a real-to-virtual Digital Shadow architecture in which the motion of a physical mobile robot is replicated by a virtual model in real time using NVIDIA Isaac Sim 5.1.0. The system estimates the robot pose using an external vision system based on AprilTag markers from the tag36h11 family and transmits this information through a ROS 2 Humble communication pipeline. The data flow is unidirectional: the virtual model in Isaac Sim is updated from the physical workspace, but it does not send control commands back to the robot.

The methodology is organized into two main stages. First, an initialization stage is performed to calibrate the overhead camera with respect to the arena using fixed AprilTag markers placed at known positions. Second, during operation, the AprilTag marker mounted on the robot is detected continuously and its pose is transformed into the arena reference frame to update the virtual robot in Isaac Sim.

3.1 System Architecture

The experimental setup is composed of five main elements: the physical robot, the overhead vision system, the perception computer, the communication network, and the Isaac Sim workstation. An overview of the system architecture is shown in Fig. 1.

The physical platform is a Freenove Tank robot that moves inside a bounded 2 m $\times$ 3 m arena. Since the robot uses tracks and operates over a smooth surface, its internal odometry may be affected by slippage. For this reason, the proposed system does not rely on the robot's internal state, but uses an external visual reference instead. An AprilTag marker is placed on the upper part of the robot and is used to estimate its pose during operation.

The vision subsystem consists of an overhead Canon R50 camera with a 24.2 MP sensor and a 10 mm wide-angle lens, mounted approximately 230 cm above the arena, providing a top-view image of the workspace. This height is

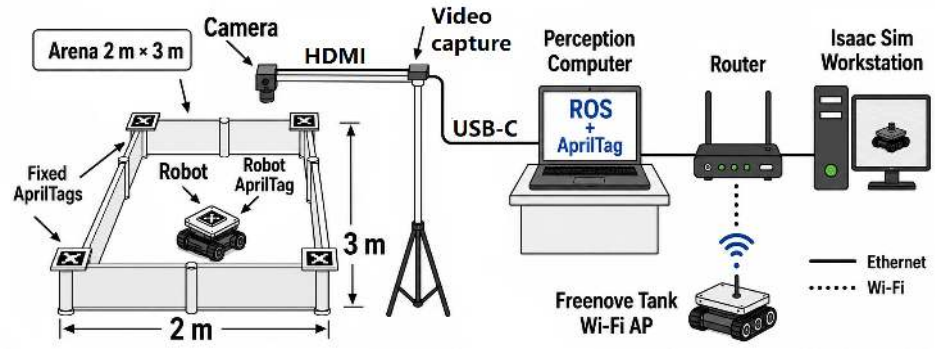


Fig. 1: System architecture of the proposed Digital Shadow setup. The Freenove Tank robot moves inside a 2 m × 3 m arena and is tracked by an overhead camera using AprilTag markers. The estimated pose is processed in ROS 2 and transmitted through the local network to update the virtual robot in NVIDIA Isaac Sim.

recalculated each time the system is calibrated. The camera stream is captured using an AVerMedia 4K HDMI Capture Card and processed as a 4K video stream with a resolution of 3840 × 2160 pixels, NV12 format, and 30 fps. The perception computer runs ROS 2 Humble and an AprilTag detection pipeline based on the OpenCV 4.13.0 implementation. The achieved processing frequency is 30 fps, limited by the camera acquisition rate. The estimated pose is then sent through the local network to a separate workstation, which runs NVIDIA Isaac Sim 5.1.0 and updates the digital representation of the robot. Figure 2 illustrates the same robot position from three different perspectives: an external view of the physical arena, the overhead camera view, and the corresponding final pose in the Isaac Sim environment.

Fig. 2: Robot position viewed from outside the arena (left), from the overhead camera(middle), and in its final position in Isaac Sim (Right).

This distributed structure separates perception, communication, and simulation tasks. As a result, the robot controller remains independent from the synchronization process, while the virtual scene in Isaac Sim is continuously updated according to the observed motion of the real platform.

3.2 Camera Calibration

The arena is defined as the global reference frame of the system. To express the robot pose in this frame, the pose of the overhead camera must first be estimated. This calibration is performed at system startup using fixed 7.8 cm AprilTag markers placed at known positions in the arena.

During this stage, the camera detects the fixed AprilTag markers and associates their 2D image coordinates with their known 3D positions in the arena. These correspondences are used to estimate the camera-to-arena transformation by solving a Perspective-n-Point problem. The resulting transformation, including the effective camera height, is stored and reused during operation, avoiding the need for continuous recalibration.

3.3 Robot Pose Estimation

Once the calibration has been completed, the system tracks the AprilTag marker attached to the robot. The AprilTag pose is first estimated with respect to the camera frame and then transformed into the arena frame using the stored calibration:

$$^{A}\mathbf{T}_{R} = {}^{A}\mathbf{T}_{C} \cdot {}^{C}\mathbf{T}_{R}$$

where A, C, and R denote the arena, camera, and robot AprilTag reference frames. The transformation $^{A}\mathbf{T}_{R}$ represents the robot AprilTag pose expressed in the arena frame, obtained by combining the camera-to-arena transformation $^{A}\mathbf{T}_{C}$ with the AprilTag pose estimated in the camera frame, $^{C}\mathbf{T}_{R}$.

Since the robot moves on a flat surface, its motion is modeled as planar. Therefore, only the horizontal position and yaw angle are used to update the virtual robot in Isaac Sim. Roll, pitch, and vertical displacement are constrained according to the known geometry of the robot and the AprilTag placement, which reduces noise and improves the stability of the Digital Shadow.

3.4 Synchronization Workflow

During operation, the system follows a simple synchronization workflow. The overhead camera detects the robot AprilTag, the perception computer estimates its pose, and the stored calibration transforms this pose into the arena reference frame. The resulting planar pose is then transmitted through ROS 2 to the Isaac Sim workstation, where the virtual robot is updated. The ROS 2 operates at 30 Hz, matching the real acquisition frequency of the camera stream.

This workflow allows the virtual model in NVIDIA Isaac Sim to continuously reflect the observed state of the physical robot. Since synchronization is based on external AprilTag-based perception rather than internal odometry, the system provides a lightweight and decoupled solution for real-time physical-to-virtual monitoring.

4 Experimentation

To validate the proposed lightweight Digital Shadow architecture, we evaluated the system's performance across two dimensions: the end-to-end physical-to-virtual synchronization latency and the accuracy and robustness of the vision-based spatial tracking.

4.1 End-to-End Synchronization Latency

In distributed architectures, synchronization latency is a fundamental metric for real-time applications. Rather than measuring a simple network ping, we evaluated the complete software pipeline latency. This metric was recorded from the exact moment the overhead camera captures a physical frame, until the resulting calculated pose is received and updated by the simulation script within the NVIDIA Isaac Sim environment. To ensure temporal accuracy across the distributed nodes, all system clocks were strictly synchronized using the Network Time Protocol (NTP).

As detailed in Fig. 3, isolating the pipeline from mechanical delays yields a consistently low total latency of 86.9 ms (standard deviation of 11.6 ms). Decomposing this end-to-end metric reveals that vision processing (AprilTag capture and estimation) averages 42.7 ms, ROS 2 network transport requires merely 0.3 ms, and the Isaac Sim physics and rendering update consumes 43.9 ms. These results demonstrate that the decoupled architecture effectively mitigates bottlenecks, ensuring highly responsive, near real-time physical-to-virtual replication.

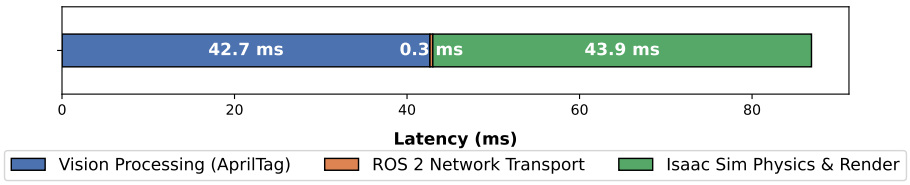


Fig. 3: System latency breakdown of the Digital Shadow update pipeline (average total: 86.9 ms), isolating software processing from mechanical response times.

4.2 Spatial Tracking Accuracy

The spatial fidelity of the virtual entity depends on the tracking capabilities of the overhead vision system. To evaluate this, the architecture was analyzed by comparing the performance of a 15 cm AprilTag marker configuration against a 7.8 cm marker across a grid of 35 known physical reference points. At each point, 10 consecutive pose predictions were recorded to measure both the measurement accuracy (Root Mean Square Error, RMSE) and the measurement dispersion (σ, representing the statistical dispersion of the pose across the 10 consecutive frames) for both translation and rotation vectors.

As summarized in Table 1, the larger marker yields highly bounded error values. Initial raw measurements revealed a nearly constant spatial offset across all 35 reference points in both the X and Y axes. This offset originates from a systematic global misalignment during the initial extrinsic calibration between the camera's reference frame and the physical ground truth. Because this error is constant across the entire workspace, it acts merely as a rigid-body translation that does not compromise the behavioral fidelity of the virtual model and can be trivially resolved via a basic coordinate transformation.

The spatial distribution of these adjusted metrics is analyzed via the heatmaps in Fig. 4. The translation accuracy heatmap demonstrates that, after correcting this constant frame shift, the residual tracking error is minimized near the center of the arena and increases marginally toward the boundaries. This spatial degradation is a known phenomenon in external vision systems, typically attributed to a combination of residual, uncompensated lens distortions at the edges of the field of view and minor inaccuracies in the camera's intrinsic calibration matrix. Similarly, the orientation accuracy demonstrates structural stability across the grid.

The measurement dispersion heatmaps further confirm the system's operational robustness. Position dispersion remains highly bounded ($\sigma = 1.8$ mm), while orientation dispersion drops to a steady state ($\sigma = 0.31°$). This spatial consistency proves that the external vision system provides the stable, non-oscillating ground truth essential for real-time physical-to-virtual state replication.

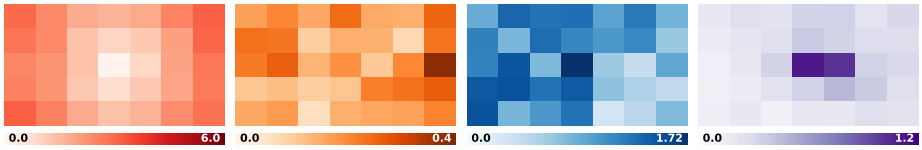


Fig. 4: Spatial performance topology for the 15 cm AprilTag marker configuration. From left to right: Position RMSE (cm), Position Std. Dev. (cm), Orientation RMSE (degrees), and Orientation Std. Dev. (degrees).

Table 1: Pose Experimental Results for the Digital Shadow Architecture

Category	**Metric**	**Small (7.8 cm)**	**Large (15 cm)**
Translation (2D Euclidean)	Mean Error	3.31 cm	2.01 cm
	RMSE	3.49 cm	2.14 cm
	Dispersion (σ)	0.99 cm	0.18 cm
Orientation (θ Yaw)	Mean Error	2.4981°	1.0218°
	RMSE	4.9123°	1.0826°
	Dispersion (σ)	8.82°	0.31°

5 Conclusion and Future Work

This paper presented a lightweight Digital Shadow architecture that synchronizes a physical mobile robot with a virtual model in NVIDIA Isaac Sim using an external vision system and AprilTag markers. Experimental validation confirms the practical viability of this decoupled approach. Despite distributing the perception and rendering pipelines across separated workstations due to physical space limitations and computational constraints, the system maintained a highly responsive average software pipeline delay of 86.9 ms. This provides a near real-time synchronization suitable for continuous monitoring without relying on heavy onboard processing or error-prone internal odometry.

Spatially, the external tracking proved highly reliable, particularly with the larger 15 cm marker, delivering a stable positional ground truth with minimal standard deviation between consecutive readings. While a slight accuracy degradation was observed near the arena boundaries—a standard consequence of lens characteristics and calibration scaling—the overall structural stability fully supports a functional digital replica. Future work will focus on transitioning this unidirectional architecture into a fully bidirectional Digital Twin to allow simulated algorithms to control the physical robot, and integrating a multi-camera setup to eliminate peripheral distortions and ensure uniform spatial accuracy across the entire workspace.

References

1. Collins, J., Chand, S., Vanderkop, A., Howard, D.: A review of physics simulators for robotic applications. IEEE Access **9**, 51416–51431 (2021)
2. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J., Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition **47**(6), 2280–2292 (2014)
3. Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B.: Characterising the digital twin: A systematic literature review. CIRP Journal of Manufacturing Science and Technology **29**, 36–52 (2020)
4. Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W.: Digital twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine **51**(11), 1016–1022 (2018)
5. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022)
6. Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P., Savatier, X.: A study of vicon system positioning performance. Sensors **17**(7), 1591 (2017)
7. Olson, E.: Apriltag: A robust and flexible visual fiducial system. In: 2011 IEEE International Conference on Robotics and Automation. pp. 3400–3407. IEEE (2011)
8. Romero-Ramírez, F.J., Muñoz-Salinas, R., Medina-Carnicer, R.: Speeded up detection of squared fiducial markers. Image and Vision Computing **76**, 38–47 (2018)
9. Wang, J., Olson, E.: Apriltag 2: Efficient and robust fiducial detection. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4193–4198 (2016)

Autonomous Grasp Dynamics and Incipient Slip Compensation using High-Resolution Tactile Sensors*

Lucas Fuente-Rodriguez[0009−0005−6743−8343], Francisco J. Rodríguez-Lera[0000−0002−8400−7079], Camino Fernández-Llamas[0000−0002−8705−4786], and Alexis Gutiérrez-Fernández[0000−0002−3173−3720]

Robotics Group, I4 Institute, University of León, León, Spain
lfuenr00@estudiantes.unileon.es, {fjrodl,cferll,agutf}@unileon.es

Abstract. The robust manipulation of unknown objects in dynamic environments remains a critical challenge for autonomous physical agents. Traditional rigid grasping strategies often fail when encountering objects with variable weights, deformability, or fragile structural integrity. To address this, a closed-loop control architecture for autonomous robotic manipulation is proposed, leveraging high-resolution tactile feedback to dynamically manage continuous contact mechanics. The system is built around a robotic gripper equipped with Contactile PapillArray haptic sensors, capable of detecting localized, incipient slip events before a macroscopic loss of grasp occurs. The continuous physical mechanism of the slip is mitigated by a time and severity based control law that dynamically adjusts gripping force to restore physical equilibrium. The primary objective of this haptic-robotic platform is to establish a reliable, autonomous sensorimotor loop that adheres to strict non-linear actuation constraints. To validate the algorithm's robustness, experimental grasping tasks were conducted across rigid, deformable, and fragile objects. The trials confirmed the system's mechanical reliability, dynamic adaptability, and safe adherence to strict mathematical thresholds, establishing a validated technical foundation for deployment in complex, unstructured physical environments.

Keywords: Robotic manipulation · Haptic feedback · Tactile sensors · Closed-loop control.

1 Introduction

The integration of autonomous robotic agents into unstructured physical environments presents complex challenges in dexterous manipulation and object

* This research is part of the project EXPLICIT (PID2024-162298OB-I00) funded by MICIU/AEI/10.13039/501100011033 and by the ERDF/EU

interaction. A fundamental requirement for these systems is the ability to autonomously grasp and transport objects without prior knowledge of their physical properties, such as mass, surface friction, or structural integrity [7]. Traditional control methodologies relying on static force thresholds or visual approximations are often insufficient for dynamic tasks. Consequently, there is a growing need in the field of physical agents to develop sophisticated sensorimotor frameworks that base robotic control in real-time, tangible contact phenomena.

To address this operational gap, autonomous manipulation strategies are increasingly incorporating high-fidelity haptic feedback methodologies. By facilitating interactive, closed-loop sensory processing, robotic agents can mimic the dynamic, reactive grip adjustments observed in biological systems [3].

This paper presents a control architecture designed to resolve the continuous dynamic mechanics of incipient slip during autonomous robotic manipulation. Specifically, we introduce a tactile processing module centered on a robotic gripper equipped with advanced haptic feedback technology, using Contactile PapillArray tactile sensors [4]. This system is tasked with a fundamental yet complex operation for physical agents: the autonomous grasping and lifting of objects, dynamically adjusting to varying load disturbances and surface conditions.

The core mechanism of this setup relies on a sophisticated, real-time sensory feedback loop. The programmed algorithm commands the robotic gripper to close until a predefined pressure threshold is detected, signalling an initially stable grip, at which point the lifting phase starts. A critical capability for dexterous manipulation in both biological and artificial systems [8] is the continuous monitoring of the contact surface for localized, incipient slip events. Upon detection of micro-sliding across the sensor matrix, the control system autonomously modulates the actuator, applying a calculated gripping force increment to prevent the object from dropping.

From a control systems perspective, the dynamic relationship between applied pressure, friction, and slip progression over time and space is governed by continuous contact mechanics. The robotic agent actively processes these spatial and temporal variables to maintain physical equilibrium [9]. The primary objective of this study is to present the design, mathematical framework, and technical validation of this haptic-robotic platform, establishing its viability as a robust subsystem for advanced physical agents.

The remainder of this paper is structured as follows: Section 2 outlines the background and related work concerning sensorimotor control architectures, tactile perception, and the continuous mechanics of robust grasping. Section 3 details the hardware architecture of the PapillArray-equipped gripper and the mathematical modeling of the slip-compensation algorithm. Section 4 presents the experimental evaluation of the system's operational dynamics and boundary conditions across various object types. Finally, the conclusions and directions for future work are presented in Sections 5 and 6, respectively.

2 Background

The development of robust grasping strategies for autonomous agents has historically been constrained by the limitations of sensory hardware. While early industrial grippers relied on basic binary contact sensors or rudimentary force transducers [11], these devices lack the spatial resolution required to accurately model dynamic contact interfaces. Modern tactile arrays, such as the Contactile PapillArray [4], represent a paradigm shift in robot perception [6]. By providing localized 3D force vectors and incipient slip detection across a dense sensor matrix, these devices supply the high-fidelity, spatio-temporal data necessary to close the loop on dynamic contact models in real time.

The transition from static friction to kinetic friction across a soft sensor array is a continuous process that propagates both spatially (across the sensor pads) and temporally (as the slip progresses). Modelling this physical propagation accurately is essential for robust grasping and contact mechanics [1, 2]. The autonomous modulation of gripping force based on this high-resolution feedback is a critical engineering challenge. Biological studies illustrate that humans instinctively rely on localized micro-sliding detection at the periphery of a contact area to unconsciously adjust grip force [3]. Replicating this behaviour in artificial systems requires a control architecture capable of processing complex, multivariate mathematical representations of the slip state [10, 5].

Our proposal addresses the need for reactive, closed-loop manipulation algorithms by using an advanced haptic feedback system as the primary driver for a dynamic control law. This approach allows the robotic agent to computationally process incipient slip metrics and dynamically alter its applied force to maintain physical equilibrium, providing a resilient solution for autonomous object handling in variable environments.

3 Materials and Methods

The experimental platform centres on a robotic manipulator integrated with Contactile PapillArray tactile sensors (see Fig. 1). This architecture is specifically designed to process high-fidelity spatio-temporal data to detect localized incipient slip events before a macroscopic loss of grasp occurs. The transition from static grip to kinetic slip across the soft, 3D sensor array is mathematically governed by a continuous distribution of frictional forces and spatial deformations. The haptic feedback is processed with a tactile sensor sampling rate of 1000 Hz.

The core of this system is the slip compensation algorithm, which employs a discrete, time-based control law to halt the continuous spatial propagation of the slip. When incipient slip is detected via the spatial deformation of the sensor array, the system dynamically computes a force increment to stabilize the object. The control law governing this required force increment, denoted as ΔF_{raw}, is tailored to the temporal and severity dynamics of the slip:

$$\Delta F_{\text{raw}} = (K_R \cdot e) + (K_T \cdot t_{\text{slip}}) + (K_I \cdot I_e) \tag{1}$$

Fig. 1. Robotic manipulator (Contactile GAL2 Gripper) equipped with the Contactile PapillArray sensors.

The proportional component ($K_R * e$) dictates the immediate reaction to the current severity of the detected slip (e). The temporal component ($K_T * t_{\text{slip}}$), acting as a linear penalization, increases the aggressiveness of the control response the longer the slip persists. Finally, the integral component ($K_I * I_e$) maintains a memory of the accumulated area of the error (I_e), preventing the system from falling short during sustained, low-magnitude slip events.

Operating with empirically tuned gains, the system optimizes its response to boundary conditions.

To ensure hardware safety and to further demonstrate the implementation of non-linear constraints in applied mathematics, the raw mathematical signal (ΔF_{raw}) is subjected to a strict actuation pipeline before being transmitted to the robotic gripper (see Fig. 2). First, the computed signal undergoes a per-cycle saturation filter to prevent erratic force spikes, clamping the saturated increment (ΔF_{sat}) strictly between 0.30 N and 4.50 N. Subsequently, this filtered increment is integrated by adding it to the current gripping force (F_{actual}). Finally, an absolute safety limit is imposed, ensuring the new commanded actuation force (F_{new}) cannot exceed a maximum threshold of $F_{\text{max}} = 30.0$ N (despite the gripper's physical capability to exert up to 40.0 N) to preserve the mechanical integrity of the tactile array.

Ultimately, this targeted mathematical response dictates the physical equilibrium, state transitions, and safe operational bounds of the autonomous grasping agent.

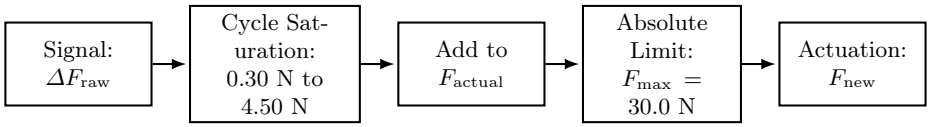


Fig. 2. Block diagram detailing the actuation pipeline for the slip compensation algorithm.

4 Experimental Evaluation

To validate the robustness of the proposed slip-compensation algorithm within the context of autonomous physical agents, a series of experimental grasping tasks were conducted. These trials were designed to verify how the underlying control model adapts to objects with significantly different physical properties, specifically focusing on rigidity, deformability, and fragility. The robotic gripper was tasked with autonomously lifting and maintaining a stable hold on three distinct categories of objects while the actuation pipeline parameters were actively monitored. Figure 3 shows the final state after grasping three different types of objects.

(a) Rigid object manipulation (hammer).

(b) Deformable object manipulation (milk carton).

(c) Fragile object manipulation (raw egg).

Fig. 3. Experimental grasping tasks demonstrating the system's adaptability to rigid, deformable, and fragile objects.

4.1 Rigid Object Grasping

The initial baseline tests used a standard rigid object (a hammer) with known mass and uniform surface friction. During these trials, the system exhibited a highly predictable response. When an incipient slip was artificially induced (e.g., by adding external downward force to the block), the proportional term ($K_R * e$) of the control law (Equation 1) dominated the initial response, providing a sharp but stable increase in gripping force until equilibrium was restored. This scenario successfully demonstrated basic closed-loop feedback, verifying the core mechanics of the controller without complex external variables.

4.2 Deformable and Variable-Mass Objects

To introduce dynamic complexity, the system was tested using a deformable milk carton under two conditions: empty and full.

When grasping the *empty* carton, the primary challenge was applying sufficient force to establish a hold without collapsing the structural integrity of the thin cardboard. The system successfully maintained a delicate grip, relying on low-magnitude adjustments that highlighted the sensitivity of the sensor array.

Conversely, grasping the *full* carton introduced significant dynamic disturbances due to the sloshing of the liquid. The movement of the fluid constantly shifted the object's centre of mass, triggering continuous, localized micro-slips across the sensor matrix. In this scenario, the vital role of the integral ($K_I * I_e$) and temporal ($K_T * t_{\text{slip}}$) components became explicitly clear. As the fluid shifted and slip persisted, the controller dynamically accumulated these error metrics, progressively increasing the gripping force over time to counteract the internal kinetic energy of the liquid, successfully stabilizing the load without human intervention.

4.3 Fragile Object Manipulation

The final experiment involved the grasping and lifting of a raw egg. This task served as the ultimate test of the system's absolute constraints and boundary conditions. The objective was to secure the egg against gravity while strictly avoiding shell fracture.

During this trial, the critical operational importance of the actuation pipeline was practically validated. While continuous slip detection naturally generated demands for increased force (ΔF_{raw}), the per-cycle saturation filter (ΔF_{sat}) successfully prevented aggressive, instantaneous force spikes that would have shattered the egg. Furthermore, the absolute limit ($F_{\max}$) ensured that the total applied force remained within a safe handling threshold for fragile objects. The successful manipulation of the fragile egg powerfully illustrated how mathematical bounds and non-linear constraints are essential for translating theoretical control signals into safe, real-world physical interactions.

5 Conclusions

This paper presented the development and technical validation of a closed-loop control architecture for autonomous robotic manipulation. By using a robotic manipulator equipped with Contactile PapillArray sensors, we successfully implemented a reactive system capable of mitigating the complex mechanics that govern incipient slip.

The experimental evaluation confirmed the robustness of our modified time-and-severity-based control law (Equation 1) across a spectrum of physical challenges. The gripper effectively maintained equilibrium when handling rigid loads, dynamically compensated for shifting centres of mass in fluid-filled containers, and safely operated within strict mathematical constraints to prevent the destruction of highly fragile objects. Crucially, these trials demonstrated that the platform responds predictably to dynamic load disturbances, confirming its viability as a robust sensorimotor subsystem. By safely and transparently managing physical boundary constraints, we have established a strong technical foundation for deploying tactile-driven physical agents in unstructured environments.

6 Future Work

Having validated the platform's mechanical and mathematical robustness, our immediate future work will focus on integrating this tactile control framework into fully autonomous mobile manipulators. Testing the algorithm during continuous navigation will introduce complex inertial disturbances, allowing us to further refine the dynamic response of the control law.

Furthermore, we intend to expand the spatial computing interfaces of the system utilizing game engines such as Unity. By streaming the high-fidelity force vectors and slip progression data into a digital twin, we can facilitate advanced teleoperation scenarios and augmented reality (AR) monitoring. Overlaying real-time, 3D visualizations of the spatio-temporal maths directly into an operator's field of view can drastically enhance the supervision and remote coordination of complex, multi-agent robotic tasks.

Declaration of AI Use During the preparation of this work, the author(s) used Google Gemini to improve the readability and grammatical structure of the manuscript. After using this tool, the author(s) thoroughly reviewed, tested, and edited all text and code. The AI was not used to generate the scientific content, and the author(s) take full responsibility for the final content and integrity of this publication.

References

1. Bicchi, A.: Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. IEEE Transactions on Robotics and Automation **16**(6), 652–662 (2000). https://doi.org/10.1109/70.897777

2. Howe, R.D., Cutkosky, M.R.: Practical force-motion models for sliding manipulation. The International Journal of Robotics Research **15**(6), 557–572 (1996). https://doi.org/10.1177/027836499601500604
3. Johansson, R.S., Flanagan, J.R.: Coding and use of tactile signals from the fingertips in object manipulation tasks. Nature Reviews Neuroscience **10**(5), 345–359 (2009)
4. Khamis, H., Albero, R.I., Salerno, M., Idil, A.S., Loizou, A., Redmond, S.J.: Papillarray: An incipient slip sensor for dexterous robotic or prosthetic manipulation–design and prototype validation. Sensors and Actuators A: Physical **270**, 195–204 (2018)
5. Komeno, N., Matsubara, T.: Incipient slip detection by vibration injection into soft sensor. arXiv preprint arXiv:2402.11879 (2024)
6. Leslie, O., Córdova Bulens, D., Redmond, S.J.: Design, fabrication, and characterization of a novel optical six-axis distributed force and displacement tactile sensor for dexterous robotic manipulation. Sensors **23**(24), 9640 (2023)
7. Li, Q., Kroemer, O., Su, Z., Veiga, F.F., Kaboli, M., Ritter, H.J.: A review of tactile information: Perception and action through touch. IEEE Transactions on Robotics **36**(6), 1619–1634 (2020)
8. Romano, J.M., Hsiao, K., Niemeyer, G., Chitta, S., Kuchenbecker, K.J.: Human-inspired robotic grasp control with tactile sensing. IEEE Transactions on Robotics **27**(6), 1067–1079 (2011)
9. Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot modeling and control. John Wiley & Sons (2006)
10. Veiga, F., Edin, B., Peters, J.: Grip stabilization through independent finger tactile feedback control. Sensors **20**(6), 1748 (2020)
11. Yousef, H., Boukallel, M., Althoefer, K.: Tactile sensing for dexterous in-hand manipulation in robotics—a review. Sensors and Actuators A: Physical **167**(2), 171–187 (2011). https://doi.org/10.1016/j.sna.2011.02.038

León Weeds: A Region-Specific Dataset for Weed Instance Segmentation

Vicente Barreiro Serrano[1][0009-0007-6932-1872],
Francisco Javier Rodríguez Lera[1][0000-0002-8400-7079],
Miguel Ángel González Santamarta[1][0000-0002-7658-8600],
Vicente Matellán Olivera[1][0000-0001-7844-9658]

Grupo de Robótica Instituto I4, EIIA Campus de Vegazana, Universidad de León, 24071 León, Spain
vbars@unileon.es

Abstract. Precision agriculture increasingly relies on deep-learning perception to enable site-specific weed management, yet annotated datasets tailored to the weed flora of specific agricultural regions remain scarce. This work introduces **León Weeds**, a publicly available dataset targeting eight weed species relevant to the agricultural *páramo* of León, Spain. León Weeds provides manually curated and polygon-level instance annotations for weed species relevant to the León agricultural region, transforming publicly available citizen-science imagery into a benchmark suitable for weed instance segmentation. The corpus comprises 1,107 images sourced from iNaturalist, filtered for quality, and annotated with polygon masks in CVAT. Combined with the Veridis crop dataset [7], it forms a 10-class, 4,359-image benchmark. The baseline YOLOv26m-seg models achieve a box/mask mAP@0.5 of 79.3%/74.5% (weed-only) and 77.5%/69.9% (weed+crop). Dataset and code are available at unileon-robotics/leon_weeds on Hugging Face.

Keywords: Weed detection · Instance segmentation · Precision agriculture · YOLO · Dataset.

1 Introduction

Weeds constitute one of the most significant biotic stresses on crop yield. Site-specific weed management (SSWM) aims to reduce herbicide use by treating only where needed [1], and deep-learning instance segmentation has emerged as a key enabler, providing per-pixel weed masks that can guide selective actuators with centimetre accuracy [2]. However, robust segmentation models require high-quality annotated data. As noted in the survey by Lu and Deng [3], existing public weed datasets—DeepWeeds [4], CottonWeedDet [5], CWFID [6]—predominantly target climatic regions not representative of the Iberian meseta, and most provide only bounding-box annotations, insufficient for instance segmentation.

This work addresses this gap with three contributions: (1) a curated, polygon-annotated corpus of eight weed species relevant to the *páramo leonés*, sourced from iNaturalist; (2) a combined weed+crop benchmark integrating León Weeds with Veridis [7]; and (3) baseline YOLOv26m-seg experiments validating the dataset.

2 Dataset Construction

The León Weeds dataset was designed to provide a publicly available benchmark for weed instance segmentation focused on species commonly found in cereal and legume crops in the León region (Spain). The dataset construction process consisted of four stages: species selection and image acquisition, quality filtering, polygon annotation, and integration with an existing crop dataset to enable crop–weed discrimination. Figure 3 summarizes the complete workflow.

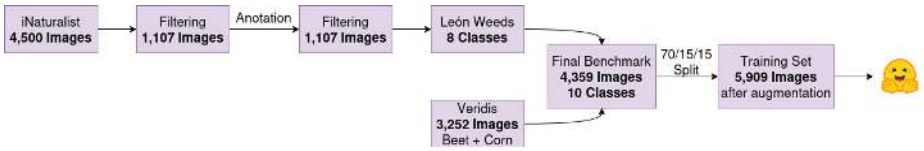


Fig. 1. Polygon instance annotation examples in CVAT on curated iNaturalist imagery.

Species selection and image sourcing. Target species were selected based on their agronomic relevance to cereal and legume crops of the *páramo* of León. Rather than acquiring field imagery, this work leverages citizen-science observations from iNaturalist [9], which aggregates geolocated, community-verified botanical photographs worldwide. A Python scraper queries the API for research-grade, openly licensed (CC-BY/CC-BY-SA/CC0) observations of nine candidate species, ranked by community agreement. Up to 500 images per species were retrieved (4,500 candidates), enabling a diverse, license-compliant corpus without field campaigns while focusing on species relevant as agricultural weeds in the León region.

Quality filtering. Each image was manually reviewed; those with insufficient plant visibility, ambiguous identity, poor quality, or non-representative viewpoints were discarded. After curation, **1,107 images** (24.6%) were retained across eight species. *Digitaria sanguinalis*, initially considered, was excluded because its fine grass-like morphology made reliable polygon delineation impractical. Table 1 shows the final distribution.

Table 1. Species distribution after filtering (1,107 images from 4,500).

Species	Imgs	Species	Imgs
A. retroflexus	314	*D. stramonium*	107
E. crus-galli	145	*S. halepense*	96
S. nigrum	140	*S. viridis*	96
C. album	129	*P. aviculare*	80

Annotation. Retained images were annotated with polygon instance masks in CVAT [10], hosted at the Universidad de León. Each visible weed instance was outlined with a tight polygon; heavily occluded instances (<20% visible) were skipped. Annotations were exported in YOLO segmentation format. Fig. 2 illustrates the process.

Integration and publication. The weed corpus was merged with the Veridis dataset [7] (sugar beet and maize crops, 3,252 images), yielding a 10-class, **4,359-image** benchmark. Stratified splitting (70/15/15) produced 3,037 train, 651 val, and 653 test images. Augmentation (flips, affine transforms, colour jitter, mosaic at p=0.3) was applied to the training split via Albumentations [11], expanding it to 5,909 images. The dataset is published on Hugging Face[1].

Fig. 2. Polygon instance annotation examples in CVAT on curated iNaturalist imagery.

3 Experiments and Results

To assess the usefulness of the proposed dataset, we trained a **YOLOv26m-seg** [8] instance segmentation model using two experimental settings. In **Exp.,1**, only weed classes were considered (8 classes), while **Exp.,2** included both weeds and crops (10 classes), enabling crop–weed discrimination.

Table 2 summarizes the performance on the test set. The weed-only configuration achieved the highest overall performance, reaching a box mAP@0.5 of 0.793 and a mask mAP@0.5 of 0.745. When crop classes were incorporated, performance decreased slightly (box mAP@0.5 of 0.775 and mask mAP@0.5 of 0.699), reflecting the increased classification complexity. Nevertheless, the combined model provides the additional capability of distinguishing crops from weeds, which is essential for precision agriculture applications.

[1] https://huggingface.co/datasets/unileon-robotics/leon_weeds

Table 2. Test-set evaluation. P = precision, R = recall.

	Box				Mask			
	mAP50	mAP50:95	P	R	mAP50	mAP50:95	P	R
Exp. 1 (Weed)	.793	.673	.859	.717	.745	.532	.857	.687
Exp. 2 (W+C)	.775	.580	.801	.700	.699	.422	.714	.636

A class-wise analysis of Exp.,2 reveals notable differences among species. *Solanum nigrum* achieved the highest performance (box/mask AP@0.5 of 0.943/0.943), likely due to its distinctive visual appearance. In contrast, *Amaranthus retroflexus* (0.597/0.570) and *Setaria viridis* (0.724/0.469) proved more challenging, mainly because of morphological variability and the difficulty of accurately segmenting thin grass-like structures. Crop classes obtained consistently high scores, with sugar beet reaching 0.923/0.905 and corn 0.906/0.797 (box/mask AP@0.5), indicating that the dataset supports reliable crop–weed discrimination.

Figure 3 presents the normalized confusion matrix for the combined model. Most classes are correctly identified, with confusion concentrated among visually similar weed species, highlighting the practical challenges represented in the dataset.

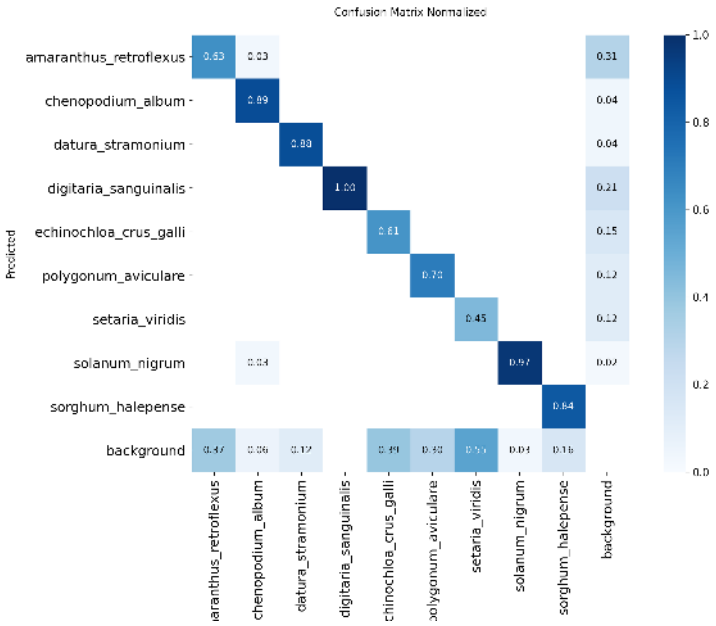


Fig. 3. Normalized confusion matrix for the combined model (Exp.,2).

4 Conclusion

We introduced León Weeds, a polygon-annotated instance segmentation dataset of eight weed species from the agricultural *páramo* of León. Using iNaturalist citizen-science imagery provides a cost-effective region-specific benchmark

without field campaigns. Baseline YOLOv26m-seg results validate its utility for edge-deployed segmentation. Future work will expand the corpus with field images and explore semi-supervised learning.

Acknowledgments. Grant AURORAS-PERMAP PID2024-161761OB-C21 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

References

1. Christensen, S., et al.: Site-specific weed control technologies. Weed Res. **49**(3), 233–241 (2009)
2. Wang, A., Zhang, W., Wei, X.: A review on weed detection using ground-based machine vision and image processing techniques. Comput. Electron. Agric. **158**, 226–240 (2019)
3. Lu, Y., Deng, B.: Weed image database development towards robust weed recognition. Update soon **Update soon**, Update soon (2023)
4. Olsen, A., et al.: DeepWeeds: A multiclass weed species image dataset for deep learning. Sci. Rep. **9**(1), 2058 (2019)
5. Chen, D., et al.: Performance evaluation of deep transfer learning on multi-class identification of common weed species in cotton production systems. Comput. Electron. Agric. **198**, 107091 (2022)
6. Haug, S., Ostermann, J.: A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks. In: ECCV Workshops, pp. 105–116. Springer (2014)
7. Sánchez de la Fuente, S., et al.: Dataset for autonomous agriculture using robots to inspect corn and beet. Data in Brief **66**, 112820 (2026)
8. Jocher, G., et al.: Ultralytics YOLO. https://github.com/ultralytics/ultralytics (2024)
9. iNaturalist. https://www.inaturalist.org, last accessed 2026/06/15
10. CVAT.ai Corporation: Computer Vision Annotation Tool (CVAT). https://github.com/cvat-ai/cvat (2024)
11. Buslaev, A., et al.: Albumentations: Fast and flexible image augmentations. Information **11**(2), 125 (2020)

Evaluating a Humanoid Social Robot and AI-Generated Adaptive Music Support in Residential Care

Esteban Caballero-Morcillo[1], Javier Ballesteros-Jerez[1], Jesus Martínez-Gómez[1], and Ismael García-Varea[1]

Computing Systems Department, Universidad de Castilla-La Mancha, Spain
Esteban.Caballero@uclm.es, Javier.Ballesteros@uclm.es, Jesus.Martinez@uclm.es, Ismael.Garcia@uclm.es

Abstract. Socially assistive robots can support physical, cognitive, and social stimulation activities in residential care, but their deployment must be assessed under real operating conditions. This paper presents a pilot evaluation of Robic, a system composed of the NAO humanoid robot, the Inrobics Clinic platform, an RGB-D camera, and a professional control interface. The system was tested in a residential care home to analyse its practical viability, user and staff response, and deployment constraints. Results suggest that Robic can support structured sessions under professional supervision, although its use depends on appropriate setup, activity selection, and staff mediation. The evaluation also revealed an opportunity to enrich repeated robot-guided activities through more personalized and motivating content, so we developed a complementary prototype that adds AI-generated adaptive musical support to Robic exercise sessions.

Keywords: Socially Assistive Robotics, Humanoid Robots, Residential Care, NAO, Human-Robot Interaction, AI-Generated Music.

1 Introduction

The ageing of the population is increasing the demand for long-term care and support in residential facilities [9, 7]. In this context, Socially Assistive Robotics (SAR) has emerged as a promising research field. These systems are not intended to replace healthcare professionals, but to complement their work in companionship, cognitive stimulation, and guided physical exercise [2, 8].

Humanoid robots such as NAO have been explored in assistive contexts due to their size, expressive movements, and multimodal interaction capabilities [4, 3]. However, using these systems in real care homes requires more than technical validation. Space constraints, staff mediation, user heterogeneity, and activity adaptation must also be considered.

This paper presents a pilot evaluation of Robic, a system based on NAO and the Inrobics Clinic platform, in a real residential care home. The goal is to analyse whether the system can support physical and cognitive activities in practice,

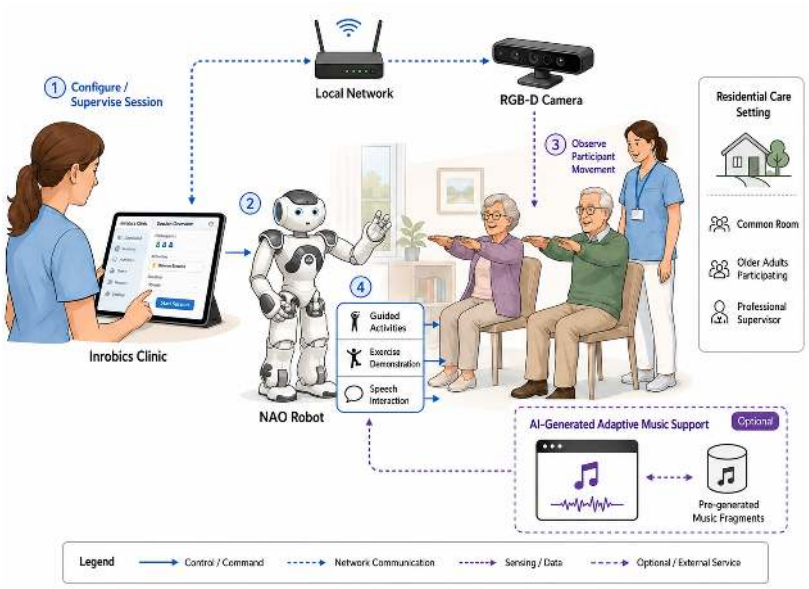


Fig. 1. Robic deployment and optional AI-generated adaptive musical support.

and under which conditions. The pilot evaluation also revealed an opportunity to enrich repeated robot-guided activities through more personalized and motivating content. Based on this observation, we developed a complementary prototype that adds AI-generated adaptive musical support to Robic exercise sessions.

The contribution of this work is therefore twofold: a real-world exploratory evaluation of Robic in residential care, and a complementary adaptive music prototype derived from the needs identified during that evaluation.

2 System Overview and Adaptive Music Support

Robic integrates three interconnected components: the Inrobics Clinic platform running on an Android device, the NAO V6 humanoid robot, and an RGB-D camera [5]. The Android device supports user and activity configuration and session supervision; NAO provides verbal instructions, gestures, and exercise demonstrations; and the camera captures body movement for monitoring. This distributed setup coordinates session configuration, robot interaction, and movement observation within the local environment.

During the evaluation, a need for greater rhythm, variety, and motivation in repeated exercise sessions was identified. To address it, a complementary web prototype was developed. MusicGen [1] was used offline to generate 30 reviewed instrumental fragments covering five low-intensity upper-limb exercises and six musical styles: rock, pop, classical, jazz, flamenco, and traditional Spanish copla. Prompts specified a clear rhythm, approximate tempo, and no vocals to avoid interfering with the robot's instructions. Professionals select an exercise and

musical style, and the Web Audio API retrieves and combines the corresponding fragments into a continuous soundtrack. The module therefore complements Robic while avoiding real-time generation delays.

3 Pilot Evaluation and Results

The pilot evaluation was conducted at the "Núñez de Balboa" residential care home in Albacete, Spain. It involved three residents and four care professionals with different roles in the centre. The professionals supported the preparation and supervision of the sessions and provided feedback on the suitability of the system for the residential care context. Activities were selected to be simple and safe, focusing on low-intensity upper-limb movements and cognitive stimulation tasks such as riddles and chained words.

The evaluation followed an iterative process. An initial visit was used to analyse the space and discuss possible activities with staff. A first demonstration was then carried out with users. Based on the observations, adjustments were made to the setup, activity selection, and role of the staff. A second demonstration was then conducted to assess the effect of these refinements.

Qualitative observations showed that the physical presence of NAO generated curiosity and attention among residents. In physical exercises, the combination of verbal instructions and visual demonstrations helped users understand the activity. The robot was also perceived by staff as a potentially useful tool for structuring sessions and encouraging participation. At the same time, the evaluation revealed practical constraints: the system required professional supervision throughout the session, correct device placement, and careful selection of exercises according to the users' abilities.

To complement the qualitative observations, usability questionnaires were administered after each iteration. Three valid user responses were collected in both iterations, while three professional responses were obtained in the first and one in the second. Both groups completed the ten-item System Usability Scale (SUS), rated on a five-point Likert scale and converted into a score from 0 to 100. Professionals also completed an Inrobics-specific questionnaire, scored from 1 to 5, focused on their perception of the system in the residential care context. The mean professional SUS score was 82.5 out of 100 in the first iteration, while the single response collected in the second iteration yielded a score of 95. Users obtained average scores of 63.33 and 70.83 in the first and second iterations, respectively. Given the exploratory nature and small sample of the pilot study, these results should be interpreted as preliminary.

The average Inrobics questionnaire score was 4.38 in the first iteration, while the single response collected in the second iteration yielded a score of 4.88 out of 5. Professionals highlighted the value of the system for cognitive stimulation and guided activities. These observations motivated the adaptive music concept as a means of adding rhythm, variety, and personalized content to repeated exercises, in line with previous work on music and exercise [6].

4 Conclusions and Future Work

This paper presented a pilot evaluation of Robic in a real residential care home and a complementary prototype for AI-generated adaptive musical support. The results suggest that Robic can support structured physical and cognitive activities, generate interest among residents, and provide a useful resource for professionals. However, successful deployment depends on adequate preparation of the environment, careful activity selection, and continuous staff mediation.

The proposed music prototype is designed to enrich robot-guided exercise sessions through reviewed offline fragments selected according to exercise type and musical style, avoiding real-time generation delays.

Future work will focus on validating this adaptive musical support in real Robic sessions using subjective and objective interaction indicators, and on expanding the audio library to include more exercises and styles.

Acknowledgments. This work has been supported by grant PID2022-137344OB-C33, funded by MICIU/AEI/10.13039/501100011033 and by "ERDF/EU". It is also framed within a collaborative project between JCCM and UCLM.

References

1. Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., Défossez, A.: Simple and controllable music generation. In: Advances in Neural Information Processing Systems. vol. 36, pp. 47704–47720 (2023)
2. Feil-Seifer, D., Matarić, M.J.: Defining socially assistive robotics. In: Proceedings of the 9th International Conference on Rehabilitation Robotics. pp. 465–468. IEEE (2005)
3. Filippini, C., Perpetuini, D., Cardone, D., Merla, A.: Improving human-robot interaction by enhancing nao robot awareness of human facial expression. Sensors **21**(19), 6438 (2021)
4. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of nao humanoid. In: Proceedings of the IEEE International Conference on Robotics and Automation. pp. 769–774. IEEE (2009)
5. Inrobics Social Robotics, S.L.: Inrobics Rehab Clinic: Manual del usuario (2026), edición 9.0
6. Karageorghis, C.I., Priest, D.L.: Music in the exercise domain: A review and synthesis. International Review of Sport and Exercise Psychology **5**(1), 44–66 (2012)
7. OECD: Who Cares? Attracting and Retaining Care Workers for the Elderly. OECD Publishing, Paris (2020)
8. Pu, L., Moyle, W., Jones, C., Todorovic, M.: The effectiveness of social robots for older adults: A systematic review and meta-analysis of randomized controlled studies. The Gerontologist **59**(1), e37–e51 (2019)
9. World Health Organization: World Report on Ageing and Health. World Health Organization, Geneva (2015)

Evaluating Context-Aware Conversation with a Mobile Social Robot in Residential Care

Esther Almendros-Saelices[1], Javier Ballesteros-Jerez[1], Jesus Martínez-Gómez[1], and Ismael García-Varea[1]

Computing Systems Department, Universidad de Castilla-La Mancha, Spain
{esther.almendros, javier.ballesteros, jesus.martinez, ismael.garcia}@uclm.es

Abstract. Mobile social robots can provide support in residential care by combining autonomous navigation, multimodal interaction and personalised activities. This paper presents an exploratory evaluation of an interaction-oriented platform for Temi, aimed at supporting accompaniment, communication and daily engagement with older adults in a care home. The system allows care-home staff to configure resident profiles, locations and personalised assistive actions. The main interaction improvement is a context-aware conversational module based on generative language models, which adapts the dialogue to the resident profile, recent conversation history, sensitive topics and the current state of the assistive round. The proposed interaction model was evaluated during a pilot deployment in a real care home. Results showed that Temi could complete the configured rounds, coordinate navigation and interaction, and elicit favourable reactions from residents and professionals. The deployment also revealed practical requirements for robust use in residential care, especially regarding voice interaction.

Keywords: Socially Assistive Robotics · Human-Robot Interaction · Residential Care · Conversational Systems · Large Language Models

1 Introduction

Population ageing is increasing the demand for long-term care, especially in residential care homes where professionals combine assistance, supervision, stimulation and social accompaniment [4]. Recent work on socially assistive robots highlights their potential to support mobility, interaction and well-being in elderly care [2]. However, their usefulness depends on how well they fit care-home routines, user needs and physical environments [3].

This paper focuses on improving the interaction capabilities of Temi, a mobile social robot with autonomous navigation, speech interaction, a touch screen, multimedia support and telepresence, for use as a complementary tool in residential care. To evaluate these improvements in a realistic scenario, the work uses an assistive round, in which the robot visits several residents and performs actions

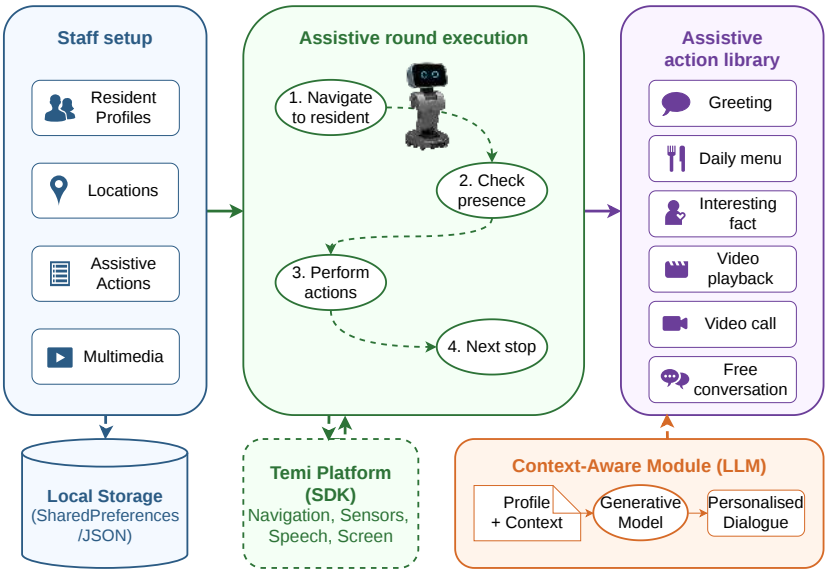


Fig. 1. Assistive round workflow used to evaluate Temi in residential care.

previously selected by care-home staff. This scenario makes it possible to analyse how navigation, presence detection, multimedia resources and personalised interaction can be integrated into a real care environment.

2 System Description

After analysing the residential care environment, we designed a native Android application running on Temi. The system supports accompaniment, information delivery, personalised stimulation, communication with relatives and conversation. Care staff can configure resident profiles, mapped locations, contacts, multimedia resources and ordered assistive rounds. Figure 1 summarises the workflow used to evaluate the platform.

Each round is organised as an ordered sequence of stops. At each stop, Temi navigates to the configured location, checks whether the resident is present, performs the selected actions and continues with the next stop. If the resident is not detected or an action cannot be completed, the system skips that interaction instead of blocking the sequence. The action catalogue includes greetings, daily menu information, personalised messages, video playback, video calls and free conversation. Most actions rely on predefined content or robot services, whereas free conversation introduces an adaptive interaction component.

3 Context-Aware Conversational Module

Free conversation is the main adaptive action of the platform. It requires the robot to maintain coherence across turns and adapt its behaviour to the resi-

dent's participation. Generative language models have recently been explored as a way to support more flexible human-robot interaction [5]. For each turn, the module builds a dynamic prompt using the robot's role and limitations, communication guidelines for older adults, the resident profile, interests and sensitive topics, recent dialogue history, and the current state of the assistive round. In this way, the response is adapted to the resident's profile, the current context and the previous conversation.

To avoid delegating full dialogue control to the language model, the response is requested in a structured format containing the verbal response, the detected conversational mode and a decision about whether the conversation should continue. The application validates this output and uses local rules to update the next prompt, adapt the interaction style, reorient sensitive topics or close the conversation before continuing with the round. Unlike a standalone chatbot, the model is embedded in a staff-configured physical workflow, keeping the interaction flexible but bounded by the assistive round.

4 Pilot Evaluation

The prototype was evaluated in a residential care home in Albacete, Spain, involving three residents and four professionals across four sessions. The aim was to analyse whether Temi could operate as a complementary support tool in a realistic environment (see Figure 2).

Group	n	Score	Rating
Residents	3	61.0	Marginal
Professionals	4	73.8	Acceptable

Fig. 2. Pilot study: interaction example and SUS scores.

During the complete pilot rounds, Temi reached the configured stops and executed the assigned actions without blocking the session flow. This showed that navigation, presence checking, action execution and round continuation were correctly coordinated in the tested environment. Predefined actions helped structure the visit through resident-specific content, such as spoken messages, multimedia resources and video calls. The main interaction improvement was free conversation. In favourable cases, the robot maintained fluent and coherent conversations across several turns, introduced topics related to the resident profile and adapted its responses to the recent dialogue context. The conversational logs also showed that the module could support personalised dialogue while keeping the interaction bounded by the structure of the round.

The deployment revealed several practical considerations for future use. We observed that while predefined actions were essential for maintaining the structure of the visit, the free conversation module naturally became the most adapt-

able form of interaction. To manage this flexibility, contextual prompting and structured responses helped keep the dialogue personalised but bounded by the round. The main limitation encountered was voice interaction: low voice volume, short responses and speech difficulties affected some conversations, although they did not prevent the robot from completing the physical round. SUS questionnaires [1] showed a positive overall perception, especially among professionals.

5 Conclusions and Future Work

This paper presented an exploratory evaluation of Temi as a mobile social robot for residential care, focusing on context-aware conversation within personalised assistive rounds. The proposed framework differs from standalone conversational systems by integrating the language model into a supervised physical workflow configured by care-home staff. Given the small number of residents, professionals and sessions, the results should be interpreted as exploratory evidence of feasibility rather than as generalisable outcomes.

Future work will focus on improving voice interaction, extending the evaluation to more residents and developing a staff-oriented monitoring tool to record and summarise the events that occur during each assistive round. We also plan to explore more adaptive rounds in which the language model can personalise the interaction from each resident profile and propose suitable actions dynamically, instead of relying only on a predefined sequence.

Acknowledgments. This work has been supported by grant PID2022-137344OB-C33, funded by MICIU/AEI/10.13039/501100011033 and by “ERDF/EU”. It has also been supported by project "Innovación Tecnológica y Social en Materia de Robótica" funded by the European Union NextGenerationEU/PRTR within the framework of "Convenio de Colaboración entre la Consejería de Bienestar Social de la JCCM y la UCLM".

References

1. Brooke, J.: SUS: A quick and dirty usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, I.L. (eds.) Usability Evaluation in Industry, pp. 189–194. Taylor and Francis, London (1996)
2. Naseer, F., Khan, M.N., Tahir, M., Addas, A., Kashif, H.: Enhancing elderly care with socially assistive robots: A holistic framework for mobility, interaction, and well-being. IEEE Access **13**, 82698–82717 (2025). https://doi.org/10.1109/ACCESS.2025.3567331
3. Trainum, K., Liu, J., Hauser, E., Xie, B.: Nursing staff’s perspectives of care robots for assisted living facilities: Systematic literature review. JMIR Aging **7**, e58629 (2024)
4. World Health Organization: Ageing and health. https://www.who.int/news-room/fact-sheets/detail/ageing-and-health (2025), accessed: 2026-07-21
5. Zhang, C., Chen, J., Li, J., Peng, Y., Mao, Z.: Large language models for human-robot interaction: A review. Biomimetic Intelligence and Robotics **3**(4), 100131 (2023). https://doi.org/10.1016/j.birob.2023.100131

Self-Extending Causal-Semantic Memory for Grounded and Self-Correcting Knowledge-Driven Planning in Cognitive Robots

Javier Ballesteros-Jerez[1], Jesus Martínez-Gómez[1], and Ismael García-Varea[1]

Computing Systems Department, Universidad de Castilla-La Mancha, Spain
{javier.ballesteros, jesus.martinez, ismael.garcia}@uclm.es

Abstract. This thesis investigates how a self-extending causal-semantic memory can ground and verify the knowledge-driven generation increasingly used by cognitive robots. The memory is built from reused ontologies, generation is grounded on it and checked before use, and the memory extends itself with little supervision. Growth follows two properties: generality, the robot finding its own capabilities, and evolution, integrating new knowledge without degrading prior competence. The work builds on the author's already-evaluated causal validator.

Keywords: Semantic memory · Causal reasoning · Cognitive robotics.

1 Background and motivation

Mobile and social robots increasingly operate in dynamic, human-populated environments with unanticipated variability. In architectures such as CORTEX, they organise world models around a semantic memory of entities, events, and relations, and must decide whether an unexpected change integrates coherently or reveals a knowledge gap. Ontologies and knowledge graphs offer a formal, queryable substrate, yet must be reused and kept tractable in real time, and may not explain genuinely new events. Large language models reason fluently but cannot be trusted unverified, which motivates integrating them in a controlled way through graph retrieval-augmented strategies. The problem this thesis addresses is how to make such behaviour trustworthy, so that generated content stays grounded in what the robot knows, is checked before it acts, and can grow from experience with little supervision.

2 Research questions and objectives

The central question is how such a memory can ground and self-correct knowledge-driven generation so that robots work in real time, detect inconsistencies, and improve from experience under little supervision. It breaks into four questions: compact memories from reused ontologies that keep the causal and event structure needed for runtime validation (**RQ1**); retrieving causal knowledge to ground

generation and planning while bounding hallucination (**RQ2**); checking any change, extension, or plan for causal admissibility before it is committed (**RQ3**); and self-extending under little guidance, so the robot finds its own capabilities (generality) and consolidates knowledge without degrading prior competence (evolution) (**RQ4**).

3 Proposed approach

The methodology forms a closed loop around the causal-semantic memory, reusing the author's preliminary validator as its entry substrate. A knowledge graph extends a reused ontology with causal relations, event evidence, and mission-level abstractions, pruned offline to stay compact. Given a goal or an observed discrepancy, retrieved subgraphs keep the LLM within what the robot knows, and its output is verified before use, since event evidence must support the change before any proposal is re-checked for grounding and causal admissibility. The key step is to make these gated updates a lifelong process of self-extension: counterfactual reasoning explains failures and proposes better variants [1], a metacognitive policy decides whether to trust output or seek more evidence, and new knowledge is consolidated only when it preserves coherence and non-regression. GraphRAG and the Model Context Protocol act as enablers, and a dedicated suite measures groundedness, causal validity, and non-regression.

4 Preliminary results and work plan

Earlier work by the author covered mobile navigation and localisation and then, in a Master's thesis and a journal article under review, an ontology-guided causal validation framework for runtime semantic-memory maintenance [2]: locality-based pruning yields a compact schema (orders-of-magnitude faster reasoning at preserved coverage), a validator that accepts well-supported changes and rejects near-miss evidence, and a retrieval-grounded, schema-constrained proposal module, all working on isolated updates over a fixed schema. The remaining work then proceeds in three phases: consolidating memory construction across domains, extending grounding and verification to whole missions and LLM-based planning, and finally developing and evaluating self-extension with the full metric suite.

References

1. Pearl, J.: Causality. Cambridge university press (2009)
2. Ballesteros-Jerez, J., Martínez-Gómez, J., García-Varea, I.: Ontology-guided causal validation for runtime semantic memory maintenance under dynamic evidence. Under review (2026)

Acknowledgments. This work has been supported by grant PID2022-137344OB-C33, funded by MICIU/AEI/10.13039/501100011033 and by "ERDF/EU".

A Local Language-Model-Assisted Replanning Module for PlanSys2

Álvaro Valencia[1], Francisco Martín Rico[1], and Francisco Miguel Moreno[1]

Intelligent Robotics Lab, Universidad Rey Juan Carlos, Fuenlabrada, Spain
a.valencia.2022@alumnos.urjc.es, francisco.rico@urjc.es, franciscom.moreno@urjc.es

Abstract. Robotic plans fail when execution-time observations contradict the symbolic state the planner relies on. We present a local language-model-assisted replanning module for ROS 2 and PlanSys2 that reads execution context, proposes structured updates to the PDDL problem, and keeps formal planning and final acceptance under deterministic control. A plugin-based architecture, a constrained JSON output contract, and explicit validation barriers let the module exploit small local language models without delegating plan synthesis to them. In an initial validation on a bookstore robot demonstrator, the module produces correct state updates, model rankings shift sharply between synthetic and real prompts, and KV-cache reuse cuts prompt-processing cost enough to make repeated local calls viable on embedded hardware.

Keywords: Robot planning · PlanSys2 · ROS 2 · local language models · replanning

1 Introduction

Task and motion execution in real environments rarely matches the symbolic assumptions made at planning time. Objects move, actions fail, and perception reveals discrepancies between the world state and the PDDL problem used by the planner. Classical planners remain attractive in robotics because they provide explicit models, traceable decisions, and formally grounded plans, but require the application to keep the symbolic state synchronized with execution.

Recent work has explored the use of language models in robotic decision making and planning, including affordance-grounded high-level control [3], prompted task-plan generation [16], and language-guided plan synthesis [9,10]. At the same time, several studies argue that language models should support, rather than replace, explicit planning and verification [17,8].

Our work follows that second view. We present a complementary module for ROS 2 [12] and PlanSys2 [13] that uses small local language models to propose updates to the symbolic problem state, while leaving formal plan generation to PlanSys2 and POPF [4]. Deployment constraints common in robotics motivate the design: limited connectivity, privacy requirements, reproducibility concerns, and resource-constrained platforms.

Table 1. Role of the language model across representative approaches. The last row is the present work.

Approach	Language-model role	Execution	Posterior validation
LLM+P	NL→PDDL translation	Remote, large	Planner rejects unsolvable
SayCan	Action selection	Remote, large	Robot affordance estimator
ProgPrompt, Text2Motion	Program / plan synthesis	Remote, large	Limited or absent
This work	PDDL problem update	Local, small	Structural, then POPF

We make three contributions. First, we present a modular replanning assistant that integrates with PlanSys2 without modifying its internal components. Second, we define a constrained prompting and validation pipeline that turns free-form model output into typed, auditable proposals over PDDL state facts [14]. Third, we report an empirical evaluation on a robotic demonstrator, with model comparisons on real prompts and an analysis of prompt caching using embedded hardware.

2 Related Work

Existing approaches differ mainly in the role assigned to the language model. LLM+P [10] uses the model as a translator from natural language to PDDL and delegates search to a classical planner. SayCan [3] selects actions from a robot skill set grounded in affordance estimates. ProgPrompt [16] and Text2Motion [9] let the model generate executable programs or action sequences directly. Most of these systems place the model inside the plan-generation loop and rely on large or remotely served models. A complementary line of work argues that current models are unreliable planners but useful within a modulo framework that keeps verification external [17,8].

Our work takes the latter position to a bounded extreme (Table 1). The model proposes only a structured belief correction to the PDDL problem, which a symbolic planner (POPF [4]) then solves; it never plans or selects actions. This narrow role makes a small local model, under a closed output contract and external validation, a practical choice rather than a limitation.

3 Method

This section outlines the design and implementation of the proposed local language-model-assisted replanning module. It details the overall system architecture, the methodology for constructing prompts and enforcing a strict JSON

output contract, the validation barriers implemented for safe failure handling, and the optimizations for local inference utilizing KV-cache reuse.

3.1 System Overview

The proposed system is an external assistance layer around PlanSys2 (Fig. 1).[1] The planner, problem expert, and executor remain unchanged. A monitor node subscribes to execution events, builds a structured textual context, queries a local language-model backend, parses the response, and returns a proposal to an external integrator node. The integrator decides whether the proposal is valid for the current domain, applies the accepted changes, and requests a new plan.

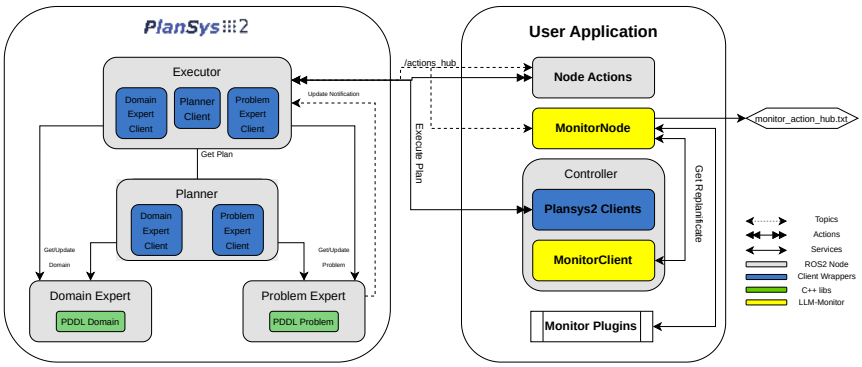


Fig. 1. Overview of the proposed architecture. The contribution of this work is the monitor node and its local inference plugin; PlanSys2 remains unchanged and the application integrator keeps the final decision over state updates.

This separation is deliberate. It avoids coupling the planner to a specific model backend and prevents the model from modifying the symbolic state directly. The language model acts as a context interpreter and hypothesis generator, not as a replacement for planning.

3.2 Context Construction and Output Contract

Each proposal request combines four information sources: the PDDL domain, the current PDDL problem, a natural-language observation from the integrator, and

[1] The module and the demonstrator are open source under Apache-2.0 at `https://github.com/plansys2-llm`.

Listing 1.1. Model response for the demonstrator state-correction case (Qwen2.5-3B).

```
{
  "classification": "MODIFY_PLAN",
  "reasoning": "pick_book failed; perception shows red_book at middle_path",
  "remove_predicates": ["(object_at red_book shelf_red)"],
  "add_predicates": ["(object_at red_book middle_path)"],
  "add_instances": [],
  "domain_changes": []
}
```

a summary of execution events captured from PlanSys2. The module compresses the event log before inserting it into the prompt. In the default *limited* mode it keeps only terminal events such as action success or failure, which reduces noise and fits the prompt within the context window of small local models.

The prompt asks the model to perform belief correction rather than planning. It forbids changes to unrelated objects and requires an exact JSON object with a closed set of classifications: `CORRECT`, `MODIFY_PLAN`, or `UNSOLVABLE`. The model expresses proposed changes as lists of predicates to remove or add, plus optional new instances. This contract turns response processing into deterministic structural parsing rather than natural-language interpretation. Listing 1.1 shows a real response from the demonstrator, where perception reports that the target book is away from its shelf.

3.3 Validation and Failure Handling

The monitor applies a first validation barrier to every answer. It extracts the first parseable JSON object, checks that the mandatory `classification` field is present and admissible, and converts the proposal into a typed ROS 2 message. The monitor maps any malformed answer to an explicit error result rather than a partial proposal. This explicit failure policy is safer than silent degradation in a robotic replanning loop.

Semantic validation remains outside the monitor. The external integrator checks whether the proposed predicates belong to the domain signature and whether referenced objects exist in the current problem instance. Only after these checks does the application update the symbolic state and trigger replanning. This split keeps the module reusable and preserves a clean trust boundary around the model.

3.4 Local Inference and KV-Cache Reuse

The reference backend relies on `llama.cpp` [6] through `llama_ros` [7] and runs small quantized models locally. Prompt ordering is a central engineering decision. We place static content first—the role, rules, output schema, and the stable PDDL domain—and volatile content last—the observation and the growing action-log summary. The subsequent requests then share a long common prefix. We modified `llama_ros` to integrate the prefix-reuse mechanism.[2] The server

[2] `https://github.com/mgonzs13/llama_ros/pull/38`

then reuses the KV cache across consecutive requests to the same persistent session and recomputes only the divergent suffix.

Prompt evaluation, not token generation, dominates latency on embedded CPUs. Reusing the cached prefix avoids recomputing the full prompt at every replanning step, turning a prompt-assembly choice into a measurable systems contribution.

4 Experimental Setup

We validate the system on a bookstore robot demonstrator: a Kobuki platform must fetch a book from a shelf and deliver it to a table. Replanning triggers when perception finds the target book at a location other than the one that the current plan assumes. The real demonstrator is a simple state-correction case. The system must remove `(object_at red_book shelf_red)` and add `(object_at red_book middle_path)`.

The broader evaluation spans three cases of increasing difficulty: a *simple* displacement, a *complex* scenario with several predicate changes, and one new instance, and an *edge* case where the state already matches perception. We grade each output on a 0–5 scale by the fraction of expected remove/add predicates it recovers: 0 when neither the classification nor any expected predicate is correct; 1 for a correct classification with no expected predicate recovered, a cap that also applies to the degenerate no-op that removes and re-adds the same predicates; 2 and 3 for partial recovery, below and above half of the expected predicates; 4 for full recovery with a wrong classification or malformed structure; and 5 for full recovery with the correct classification and valid JSON. In the edge case, where the state already matches perception, a 5 requires classifying `CORRECT` with empty update sets. Spurious predicates beyond the ground truth do not lower the score; we count them separately as hallucinations, a distinct failure mode.

We build the comparison in stages. An intermediate version uses synthetic prompts across a wide set of sub-4B models. The final version replays a real prompt captured from the demonstrator, with YOLO-based observations and recent action history. We switch to the real prompt because the synthetic benchmark fails to predict behavior on real inputs. We run GPU experiments on an Intel i7-12700H with an RTX 3050 Ti Laptop GPU, CPU-only experiments without GPU offload, and the embedded study on a Raspberry Pi 5 with 16 GB RAM.

5 Results

This section evaluates the proposed module's performance and practical viability using the bookstore robot demonstrator. It presents a comparative analysis of various language models when subjected to real versus synthetic prompts, details the processing latency measurements during embedded deployment, and discusses the system's practical implications and current limitations.

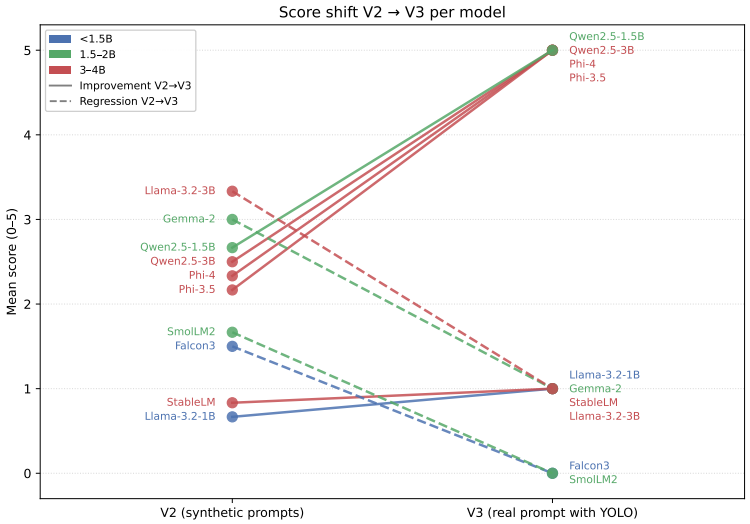


Fig. 2. Mean score (0–5) per model on synthetic prompts (left) and on the real demonstrator prompt (right).

Table 2. Representative results with the real demonstrator prompt, reported separately for the GPU and CPU-only backends. Scores follow a 0–5 rubric against ground truth; latencies are medians.

Model	Score (GPU)	Score (CPU)	Lat. GPU	Lat. CPU
Qwen2.5-1.5B [15]	5.0	1.0 (no-op)	2.0 s	6.9 s
Qwen2.5-3B [15]	5.0	5.0	3.8 s	11.4 s
Phi-4-mini [2]	5.0	5.0	5.3 s	15.8 s
Phi-3.5-mini [1]	5.0	5.0 (+3 halluc.)	5.6 s	41.7 s
Llama-3.2-3B [11]	1.0	1.0	3.9 s	11.8 s
Gemma-2-2B [5]	1.0	1.0	3.8 s	12.5 s

5.1 Real-Prompt Model Comparison

Our central finding is that model rankings change sharply between synthetic and real prompts. Under the real demonstrator prompt, only the Qwen2.5 and Phi models produce consistently correct updates. Models that rank higher on the synthetic benchmark, such as Llama-3.2-3B, often produce semantic no-ops.

Fig. 2 shows the ranking inversion directly. Llama-3.2-3B, the strongest model on synthetic prompts, drops on the real prompt to a constant `remove` = `add` pair, a semantic no-op. The Qwen2.5 and Phi families instead rise to a perfect score. Only models that correctly process the eight YOLO detections and respect the in-prompt guidelines succeed.

Table 2 summarizes the surviving models on the GPU and CPU-only backends. On GPU, Qwen2.5-1.5B dominates the Pareto frontier with a perfect score, the lowest latency, and about 1 GB of VRAM. On CPU it collapses to

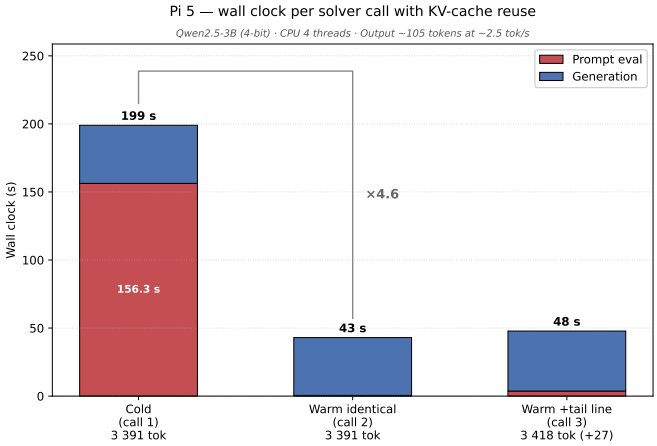


Fig. 3. Wall-clock per solver call on a Raspberry Pi 5 (Qwen2.5-3B, 4-bit, CPU). A cold call is dominated by prompt evaluation; KV-cache reuse turns the shared prefix into a marginal cost, so warm calls are dominated by generation.

a semantic no-op, while Qwen2.5-3B keeps a perfect score. Decoding is greedy (`temperature` = 0), so small numerical differences between the CUDA and CPU kernels can flip a token at a decisive step. In a 1.5B model the margin between the right token and the alternative is narrow, so that divergence surfaces as a wrong update. Phi-3.5-mini illustrates why scores must be read together with the failure-mode annotations: it keeps a perfect score on CPU because the required correction is complete, yet it adds three constant hallucinated predicates that would modify the state beyond the necessary correction, which rules it out in practice. The recommendation is therefore backend-dependent: Qwen2.5-1.5B with GPU, Qwen2.5-3B on CPU-only platforms.

5.2 Embedded Deployment

The Raspberry Pi 5 study shows the gap between functional viability and practical responsiveness. On the real prompt (≈3,400 tokens), a cold inference with Qwen2.5-3B takes about 199 s, with 156 s in prompt evaluation alone (Fig. 3). With KV-cache reuse, a repeated identical call drops to about 43 s, a ×4.6 speed-up, and a call that changes only the final prompt tokens to about 48 s, since the server recomputes only the divergent suffix. The Pi 5 stays slow in absolute terms, yet prompt evaluation turns the dominant bottleneck into a manageable cost. This shift makes a reactive replanning loop viable on this class of hardware.

5.3 Discussion and Limitations

The results support three practical conclusions. First, local small language models can be useful for symbolic state correction if they operate under a strict output contract and if a deterministic component remains responsible for final

acceptance. Second, prompt realism matters: a benchmark built from synthetic templates can select the wrong model for deployment. Third, systems-level details, such as cache reuse, can matter as much as the model choice on embedded hardware.

The evaluation also has clear limits. It centers on one demonstrator domain and one real prompt family, so these results constitute an initial validation rather than a general demonstration, and we do not generalize the ranking to unrelated tasks without re-measurement. The real-prompt experiments focus on a simple correction case; more complex real-world corrections remain future work. The evaluation also lacks a no-LLM baseline: a rule-based mapping from YOLO detections to PDDL predicates would cover the simple displacement case and is a natural baseline for future work, whereas the language model targets corrections that require contextual interpretation—multi-predicate updates, new instances, or deciding between `CORRECT` and `UNSOLVABLE`—which a fixed mapping does not address. The module validates output structure internally, yet semantic and safety validation still depend on the surrounding application.

6 Conclusion

We presented a local language-model-assisted replanning module for PlanSys2 that treats language models as proposal generators rather than planners. The module combines ROS 2 integration, constrained prompting, typed parsing, and explicit validation boundaries to keep symbolic replanning auditable. Our experiments on a bookstore demonstrator show three results: the module produces correct PDDL state corrections in the evaluated scenario, model selection depends strongly on real prompt structure, and KV-cache reuse makes repeated local inference practical on embedded platforms.

Acknowledgements. This work was supported by the European Union's Horizon Europe research and innovation programme under grant agreement No. 101070254, CoreSense: A Hybrid Cognitive Architecture for Deep Understanding. It was also supported by Grant PID2024-161761OB-C21 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

References

1. Abdin, M., et al.: Phi-3 technical report: A highly capable language model locally on your phone. arXiv preprint arXiv:2404.14219 (2024)
2. Abdin, M., et al.: Phi-4 technical report. arXiv preprint arXiv:2412.08905 (2024)
3. Ahn, M., Brohan, A., Brown, N., et al.: Do as i can, not as i say: Grounding language in robotic affordances. In: Proceedings of the 6th Conference on Robot Learning (CoRL) (2022)
4. Coles, A.J., Coles, A.I., Fox, M., Long, D.: Forward-chaining partial-order planning. In: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS). pp. 42–49 (2010)

5. Gemma Team: Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118 (2024)
6. Gerganov, G., contributors: llama.cpp. https://github.com/ggml-org/llama.cpp (2024), accessed: 2026-05-06
7. González-Santamarta, M., contributors: llama_ros: ROS2 bindings for llama.cpp. https://github.com/mgonzs13/llama_ros (2024), accessed: 2026-05-21
8. Kambhampati, S., Valmeekam, K., Guan, L., Stechly, K., Verma, M., Bhambri, S., Saldyt, L., Murthy, A.: LLMs can't plan, but can help planning in LLM-Modulo frameworks. arXiv preprint arXiv:2402.01817 (2024)
9. Lin, K., Agia, C., Migimatsu, T., Pavone, M., Bohg, J.: Text2Motion: From natural language instructions to feasible plans. In: Robotics: Science and Systems (RSS) (2023)
10. Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., Stone, P.: LLM+P: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477 (2023)
11. Llama Team, AI @ Meta: The Llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
12. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022). https://doi.org/10.1126/scirobotics.abm6074
13. Martín, F., Clavero, J.G., Matellán, V., Rico, F.J.: PlanSys2: A planning system framework for ROS2. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 9742–9749 (2021). https://doi.org/10.1109/IROS51168.2021.9636544
14. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL — the planning domain definition language. Tech. rep., AIPS-98 Planning Competition Committee (1998)
15. Qwen Team: Qwen2.5 technical report. arXiv preprint arXiv:2412.15115 (2024)
16. Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., Garg, A.: ProgPrompt: Generating situated robot task plans using large language models. In: IEEE International Conference on Robotics and Automation (ICRA) (2023)
17. Valmeekam, K., Marquez, M., Sreedharan, S., Kambhampati, S.: On the planning abilities of large language models: A critical investigation. arXiv preprint arXiv:2305.15771 (2023)

Episodic and semantic memories for causal explanation of perceptual anomalies

Pablo Bustos[1], Jorge Calderón[1], Álvaro Tardío[1], Javier Ballesteros-Jerez[2], Jesus Martínez-Gómez[2], Ismael García-Varea[2], José Galeas[3], and Antonio Bandera[3]

[1] RoboLab, Universidad de Extremadura {calde, pbustos}@unex.es
atardiof@alumnos.unex.es
[2] Computing Systems Department, Universidad de Castilla-La Mancha, Spain
{javier.ballesteros, jesus.martinez, ismael.garcia}@uclm.es
[3] Universidad de Málaga {jgaleas, ajbandera}@uma.es

Abstract. Autonomous robots often fail silently when an unexpected event breaks their predefined world model, falling back on generic exception recovery rather than understanding what happened. This paper extends the CORTEX cognitive architecture with two new memories that enable causal explanation of perceptual anomalies. The semantic memory grounds the robot's working memory into a domain ontology, validates structural changes against recorded evidence, and, when a change is unexplained, queries an LLM for a set of schema-constrained candidate causes. The episodic memory complements this with a state-restorable record of the robot's experience, providing the baseline needed to validate each candidate through internal simulation. We evaluate the approach on a person-following task in which an undetected terrain perturbation causes a transported object to fall, both in simulation and on a real platform. The resulting causal explanation is recovered consistently across three different LLM backends and validated through a single real episodic experience, without requiring large accident datasets, demonstrating a practical methodology to resolve unforeseen execution anomalies by combining generative models with internal physical verification.

Keywords: Robotics · Knowledge Representation · Ontologies · Causal Reasoning

1 Introduction

Autonomous robots that navigate dynamic environments often rely on a predefined conceptual representation of the world and a set of parametrized controllers or behaviours. However, these architectures limit their operational flexibility and capacity to adapt to unforeseen scenarios. A significant advancement would be transitioning to architectures capable of dynamically updating their world model at runtime. Dynamically extending a robot's world model during execution is an open challenge that requires bridging symbolic knowledge with causal verification.

One approach to this problem is identifying perceptual anomalies that lead to inconsistencies in the robot's working memory and reacting to them using contrastive and counterfactual reasoning tools [18, 19]. We envision this reaction as a trigger that unfolds a sequence of epistemic and pragmatic operations. Under certain conditions, these operations, result in a structural change to the concept architecture and to the control system. The resulting system now includes a new concept that can explain the anomaly and recognises the causing object whenever it appears again, and is internally interpreted in the control system to trigger an safe-preserving anticipatory action.

This work is part of the INSIGHT project [4] and builds on the CORTEX cognitive architecture [6, 3], which it extends with two new memories.

The contributions of this paper are:

A semantic memory. The design, testing and integration of a semantic memory in CORTEX that provides several functions: local ontology grounding, model checking on the working memory and anomaly detection, search of possible causes in an external, context injected LLM, and distillation of plausible explanation candidates.

An episodic memory. The design, testing and integration of an episodic memory in CORTEX that provides a permanent, efficient and real-time storage and recovery of the missions accomplished by the robot, in such a way that they can be replayed internally as a form of imagination.

A set of experiments. A perceptual anomaly is detected, analysed, and processed into an extension of the current control architecture.

2 Related Work

Modern autonomous robotic architectures increasingly incorporate structured memory systems to transition from reactive behaviours to high-level cognitive reasoning. These designs vary significantly depending on whether they target general cognitive frameworks or domain-specific application requirements, such as navigation, social interaction, or object manipulation.

Recent structural alternatives highlight how memory organization is driven by the robot's operational domain and its specific environmental challenges. For instance, frameworks tailored for social robotics and human-robot interaction, such as SUMMER [12], deploy train-free multimodal structures that selectively filter environmental data based on emotional salience and scene novelty rather than continuous logging. Conversely, architectures addressing fine-grained physical manipulation, such as Chameleon [10], introduce control-indexed prospective designs to mitigate observation-action delays during visuomotor execution. These specialized alternatives demonstrate how embodiment and task constraints shape the global architecture, contrasting with general-purpose cognitive systems.

2.1 Episodic Memory Representations

Robotic episodic memory implementations diverge significantly in their underlying data structures. Sub-symbolic encoders, such as deep autoencoders, compress continuous visual data to guide agent exploration but lack explicit symbolic tracking and suffer from catastrophic forgetting as sequential network weights are overwritten [24]. Hierarchical text-based systems utilize language models to summarize long-term experiences for human-robot interaction, though they sacrifice the exact metric accuracy needed for physical simulation [8]. Graph-based frameworks enforce spatiotemporal consistency or multi-layered hierarchies to handle dynamic environments and loop closures [20, 1].

In contrast to these approaches, our architecture combines semantic grounding with strict, state-restorable chronological checkpointing within a unified graph representation. This design enables the reliable spatiotemporal reconstruction of past states to isolate precise temporal windows and resolve perceptual anomalies via internal physics simulations [5].

2.2 Semantic Memory and Knowledge Representation

Symbolic memory systems for robotics typically rely on either static, handcrafted ontologies or fully dynamic knowledge graphs built from perception alone. Upper-level frameworks such as SOMA [2] and OCRA [15] provide rich, general-purpose vocabularies for representing actions, objects and their relations, but their breadth makes direct runtime use within real-time robotic control challenging. Other approaches couple large language models directly to the robot's belief state to interpret scenes or generate plans, gaining flexibility at the cost of weaker guarantees on the validity of the generated knowledge.

In contrast, our semantic memory adopts a hybrid strategy: an offline procedure distills SOMA and OCRA into a compact, domain-specific operational ontology, which is then grounded online into the working memory through a deterministic, event-driven mapping. This keeps symbolic reasoning within the execution bounds of mobile robotics while reserving the generative capabilities of LLMs for the bounded, schema-constrained task of proposing candidate causes, rather than for open-ended scene interpretation.

2.3 Ontological Comparisons and Semantic Memory Trade-offs

Knowledge representation in robotics typically exhibits a dichotomy between heavy, standard-driven formal ontologies and unstructured, fully dynamic graphs. Foundational architectures like KnowRob [22] (utilizing SOMA [2]) and frameworks aligned with the IEEE 1872 CORA standard [11] (such as OCRA [15]) serve as robust foundational models. They offer comprehensive, general-purpose vocabularies for abstract reasoning about actions, objects, and spatio-temporal relations. However, their expressive breadth introduces high computational complexity, making direct, real-time ontological reasoning inefficient during high-frequency robotic control loops. Conversely, recent trends bypass formal structures entirely by coupling Large Language Models (LLMs) directly to the robot's

belief state for open-ended scene interpretation and planning. While highly flexible, these neural approaches lack deterministic guarantees, leaving the robot's internal knowledge base susceptible to hallucinations or invalid state representations.

Our approach establishes a distinct operational middle ground compared to these paradigms. Instead of deploying an unconstrained upper ontology like CORA or running an extensive reasoning engine like KnowRob at runtime, we introduce a pragmatic offline reduction. Relevant domain sub-graphs from SOMA and OCRA are selected to build a compact, specialized operational ontology. Online, this distilled schema is strictly grounded into the working memory via an event-driven mapping. This hybrid architecture preserves the formal validation capabilities of established robotic ontologies within deterministic execution bounds, while constraining LLMs to a well-defined, schema-governed role: generating structured candidate causes rather than managing open-ended scene abstractions.

3 CORTEX Architecture for Causal Reasoning

CORTEX is a modular cognitive architecture built around a set of agents that cooperate through a shared, distributed representation. We refer the reader to the published descriptions and applications [3, 7, 13, 14, 9]. Here we show graphically the configuration used in these experiments and leave for the next sections a more detailed explanation of the episodic and semantic memories.

Figure 1 illustrates the overall organization of the CORTEX architecture and the interaction flow among the agents, highlighting the role of the DSR as the shared coordination layer for agent communication. The working memory (WM), at the centre, holds the robot's current belief state as a graph in which nodes represent elements in the scene and edges represent their relationships: RT edges encode geometric SE(3) poses, while the remaining edges encode logic predicates. To the left, the semantic memory holds an ontology of indoor, service-robot elements and relationships; to the right, the episodic memory stores a record of accomplished missions. At the top, a set of sub-cognitive modules provide the sensor and action streams to the WM.

Each memory lives in a different DDS domain and multicast direction, so that the semantic and episodic memory agents are dual agents, each connecting to two domains to bridge their local memory with the shared WM. Blue rectangles in the figure represent the agents that pump information among the memories.

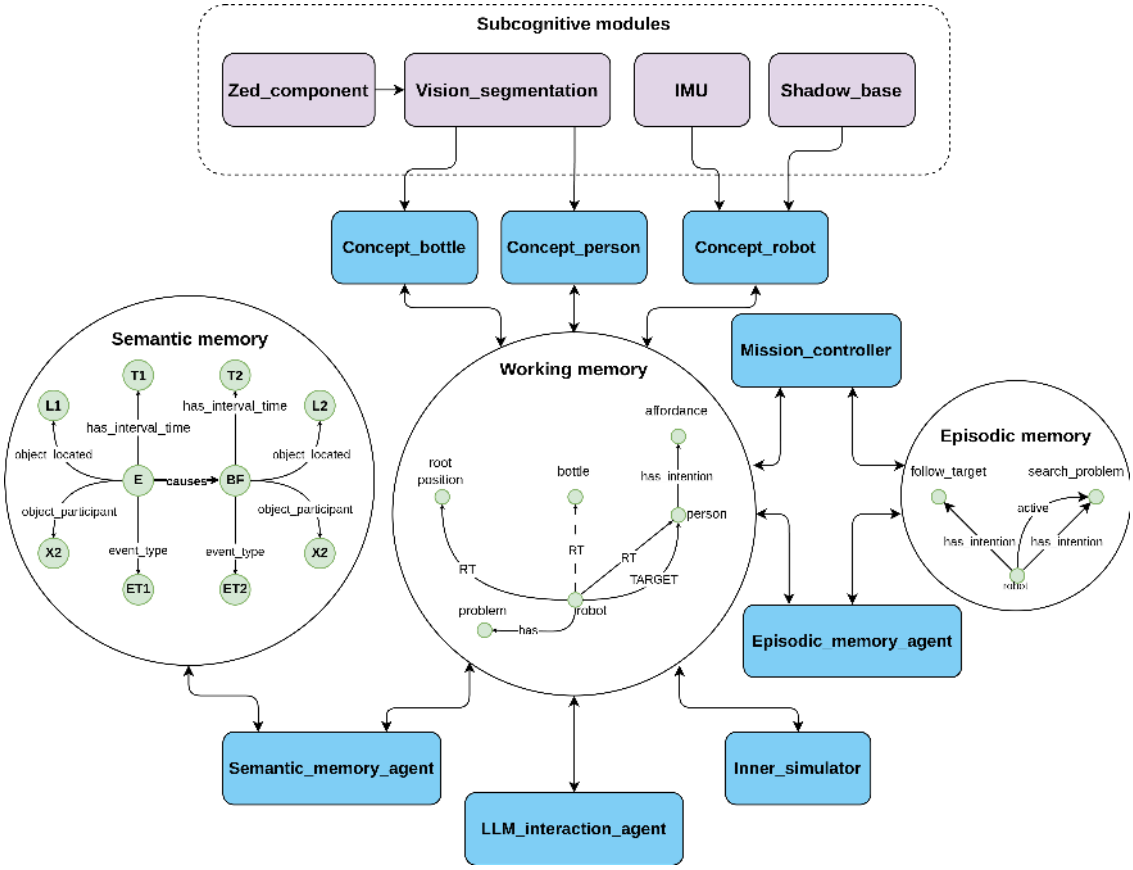


Fig. 1. The CORTEX architecture in the configuration used for the experiments presented here. CORTEX is a set of distributed memories connected by agents (blue rectangles): the working memory (WM) at the centre, the semantic memory on the left, and the episodic memory on the right.

4 Causal Explanation Methodology

The proposed methodology provides CORTEX with an inline and structured mechanism to explain perceptual anomalies as causal events, rather than treating them as isolated failures. The workflow (Figure 2) begins when the Semantic Memory Agent detects a structural inconsistency or unexpected change within the DSR. This observable discrepancy is interpreted as a non-acceptable event and triggers a causal search process over the current world model and task context.

The method for identifying inconsistencies relies on the prior occurrence of events that can justify a given structural change, which is itself caused by an event. The validation process returns a positive result when at least one pre-

viously recorded event provides valid support for the observed change. Such support requires, at a minimum, temporal, spatial, and participant compatibility. The entire process relies on a semantic memory that classifies DSR elements into different categories, including events.

From the trigger that identifies an inconsistency, the system builds a candidate-cause space and prioritizes explanations that are consistent with the observed DSR transition. In this stage, semantic constraints from the ontology guide the interpretation of the event and the formulation of plausible causes. The LLM Interaction Agent can be used as a semantic support layer to reformulate and organize candidate explanations at a higher conceptual level, bridging low-level symbolic observations with high-level explanation-oriented hypotheses. The specific criteria and mechanism for prioritizing and ranking these candidate hypotheses are detailed in Section 6.

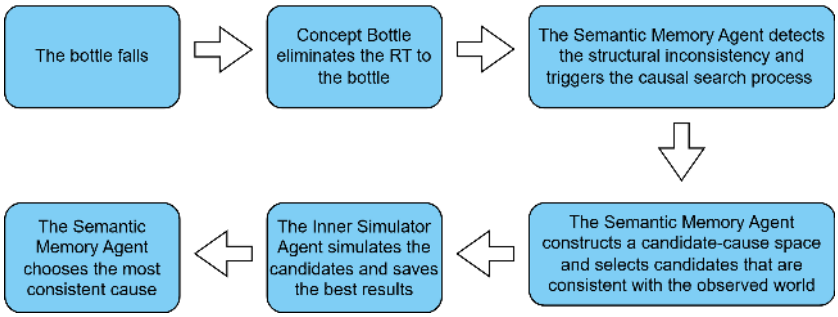


Fig. 2. Causal Explanation Workflow

After candidate generation, the Inner Simulator tries to validate them. Each candidate cause is instantiated as a synthetic scenario and executed to produce simulated Inertial Measurement Unit (IMU) traces. These traces are then temporally aligned with episodic-memory records from the real mission and compared by similarity, yielding a ranked set of hypotheses. The contrasted cause is selected from this ranking as the explanation that best matches both physical simulation and recorded experience, enabling CORTEX to produce a grounded causal account of the anomaly to safely adapt the robot's subsequent behaviour.

5 The Episodic Memory and multiple-connected agents

The Episodic Memory Agent (EMA) is a dedicated module within CORTEX responsible for recording, compressing, and recovering specific historical states of the robot's spatial working memory. The EMA captures DSR state changes, serializing the data into an in-memory database.

5.1 Conceptual Foundations

The conceptual architecture of our episodic memory is grounded in cognitive principles that distinguish context-independent factual knowledge from contextually bound, spatiotemporally indexed experiences [23]. Rather than simply logging passive telemetry, the EMA maintains an unbroken chronological record of specific state configurations, metric coordinates, and topological relationships, capturing the structural evolution of the robot's subjective experience [4]. Functionally, this continuous historical repository acts as an enabling substrate for causal reasoning, aligning with interventionist theories of causation [18, 19]. When the robot encounters an unexpected execution anomaly, it cannot synthesize a causal explanation using its world model alone. Instead, it requires a mathematically and restorable baseline to simulate counterfactual interventions, isolate the physical mechanism of failure and internalize new causal rules [5].

The EMA implementation has evolved from its foundational architecture, described in [7], which relied on a preliminary design that serialized graph states into flat, independent JSON structures. The current design introduces a serialization framework that preserves the temporal and structural history of the environment. It records time-stamped key-frames (graph state snapshots) alongside structural and attribute modifications within the DSR. This transforms the agent from a passive diagnostic log into an active retrospective mechanism capable of recovering historical context, tracing cross-agent causality, and establishing the baseline configurations required to seed internal physics simulations.

5.2 Differential Encoding and Incremental Memory Workflow

The EMA workflow ensures mathematical reproducibility with minimal storage overhead by combining full-state base key-frames with a sequence of directional delta vectors that encode only the incremental changes between graph states. The agent asynchronously monitors the DSR working memory, utilizing a two-phase logging mechanism. Upon initiating a nominal mission phase, the system records a base key-frame, which serializes the full topological, semantic, and metric state of the spatial graph, augmented with synchronized sub-cognitive sensor readings such as IMU trajectories. For all subsequent timesteps, the EMA switches to differential mode, computing and storing only directional delta vectors, which define the minimal set of atomic operations required to transform the preceding graph state into the current one. These deltas are saved sequentially into a timestamp-indexed database, creating an event-driven historical log that guarantees the exact reconstruction of the robot's world model.

5.3 Temporal and Causal Retrieval

The database supports two recovery methods: temporal state restoration, which deterministically reconstructs the world state at an exact timestamp by fetching the closest preceding base key-frame and applying the ordered delta vectors; and

structural delta queries, which search the delta vectors directly for specific structural changes (e.g., the sudden deletion of a tracked object's relational edge). These methods isolate the exact temporal window of an anomalous event, restoring the graph to seed the internal simulator and validate causal hypotheses [5].

6 The Semantic Memory: Ontologies and LLMs

The Semantic Memory Agent provides the symbolic and conceptual structure required to interpret changes in the DSR. It continuously grounds the content of the DSR into a domain ontology, maintains a queryable mirror of that ontology in a knowledge graph, and evaluates whether each observed structural change is causally admissible. When a change cannot be justified with the available knowledge, the agent escalates it to the hypothesis-generation pipeline described in Section 6.4.

The conceptual knowledge builds on the SOMA [2] and OCRA [15] foundational frameworks, which an offline procedure merges and reduces to a compact operational ontology that preserves the entities and causal properties relevant to the use case, keeping symbolic reasoning within the execution bounds of mobile robotics. This paper addresses its runtime side, detailed in the following subsections.

6.1 Ontology Grounding

The agent maintains a deterministic mapping between DSR elements and a set of RDF triples expressed in the operational vocabulary. The managed signature is intentionally small: it covers the entities relevant to the mission (the room, the robot, the followed person, the transported bottle, the follow action and an unexplained-intention marker) and the relations needed to relate them (*rdf:type*, *hasLocation*, *hasParticipant* and *hasIntention*) aligned with the DUL/SOMA upper-level categories *PhysicalAgent*, *PhysicalObject*, *PhysicalPlace* and *Action*.

Grounding is event-driven. The agent subscribes to the DSR change notifications through node and edge insertions, updates and deletions, and, for each structural change, recomputes the affected facts. For instance, an RT edge linking the robot to the bottle is translated into the assertion that the bottle is located on the robot; when that edge is later attached to the room, the previous location fact is retracted and the new one asserted. This synchronisation keeps the symbolic state consistent with the belief state at the cost of a bounded and local computation per event. Figure 3 shows the resulting operational state for the nominal mission phase, where ten triples over this signature are enough to capture the entire scene.

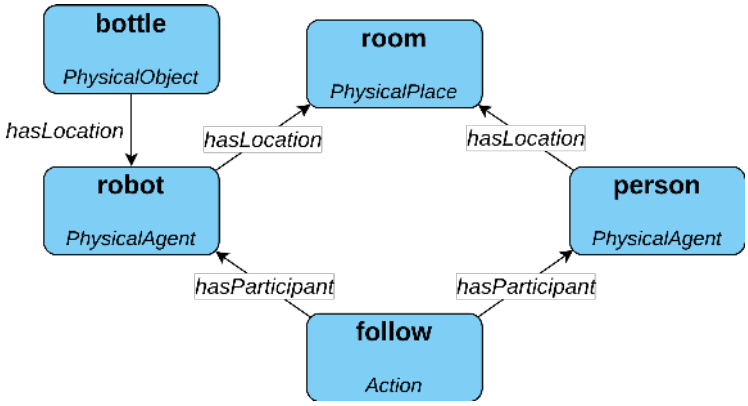


Fig. 3. Operational semantic state grounded from the working memory during the nominal mission phase, drawn as an RDF graph. Each node is an *insight:* individual with its *dul:* class shown beneath it (the *rdf:type* assertions), and labelled arrows are the *dul:* object properties. The whole scene is captured by ten triples; the anomaly studied in the experiment is the single change of the bottle's *hasLocation* from *robot* to *room*.

6.2 Knowledge-Graph Synchronisation

The operational assertions are mirrored, in real time, into an external GraphDB triple store under a dedicated named graph [17]. On start-up, the agent bootstraps its local snapshot from the remote store, so that it resumes from the last persisted state rather than re-publishing the whole graph. From then on, every grounding update is reduced to a delta of added and removed triples and pushed to the store as a single SPARQL update over the graph. The result is a persistent, queryable knowledge graph that remains eventually consistent with the working memory while respecting the real-time constraints of the architecture.

6.3 Causal Validation

A causal validator inspects each state delta and decides whether the transition is self-explanatory or requires further reasoning. It follows a validation-before-generation policy: a change is accepted only when memory already contains evidence that supports it, and is otherwise flagged as unexplained. The supporting evidence must be compatible with the observed change along several dimensions: type, time, location, participants and mechanism.

This policy is instantiated for the mission-critical precondition of the experiment: the transported bottle must remain on the robot. When the corresponding location fact is retracted without an admissible supporting cause, the validator marks the transition as unexplained and records the triggering delta. This makes the epistemic gap explicit and observable to the rest of agents, halts the mission, and opens the hypothesis-generation stage.

6.4 Hypothesis Generation

When a causal validator cannot explain a structural change, the Semantic Memory Agent assembles a context package around the anomaly and queries an LLM for a set of candidate causes. In contrast to free-form prompting, the stage is constrained on both ends: the input is grounded in the current symbolic state and in a model of the robot, and the output must conform to a strict schema before it is accepted. The generative component thus remains subordinate to the symbolic layer, which keeps control over what can be proposed and validated.

Each query requests a fixed number of hypotheses organised into two families: internal and external. Internal hypotheses attribute the anomaly to the robot's own subsystems (such as a suspension oscillation or an abrupt lateral manoeuvre), whereas external hypotheses attribute it to a local physical change in the environment (such as a collision with unmapped objects). Every candidate is returned as a JSON object carrying an explicit rationale, a confidence score, a list of grounding anchors, the expected observations and a simulation blueprint to reproduce the hypothesised cause in the Inner Simulator. Batches that fail to comply with the output schema (incorrect number of hypotheses per family, missing fields, or external hypotheses without an environmental asset) are rejected, so that only well-formed candidates reach the validation loop.

To keep the proposals physically plausible, the prompt is conditioned on a textual self-model of the robot that declares the robot identity, physical structure, locomotion, sensing and compute, together with explicit limits. In each query, a snippet of the robot's description (in Markdown format) is included, so that the internal hypotheses are limited to the subsystems that the robot actually possesses. This allows the generated internal causes to refer to concrete components of the robot instead of generic or non-existing hardware or facts.

Generation is served through Ollama [16], which exposes both cloud-hosted and locally-run models behind a single interface. The agent is configured with a primary cloud model and a local fallback, and it retries across two client transports, so that a transient failure of one backend does not abort the explanation process. The complete context package (triggering delta, current triples, inventory of DSR nodes and the self-model excerpt) is assembled, sent, decoded and schema-validated. The accepted batches are persisted with provenance metadata such as the model used, the client, the elapsed time and the token count.

7 Experimental Validation: Person Following with Obstacle

In the proposed experiment, the robot Shadow tracks a person while carrying a bottle on its tray. While performing the tracking task, the robot traverses a terrain irregularity, which induces a perturbation and results in the bottle falling off the tray.

7.1 Robotic Platform: Shadow

Shadow (see Fig. 4) is equipped with a wide range of sensors and tools that are useful in the fields of social robotics and navigation, including:

- Two 360° LiDAR sensors: a RoboSense Helios 32 mounted on top and a RoboSense Bpearl mounted on the lower part of the platform.
- A Ricoh Theta Z1 360° camera.
- A ZED 2i RGB-D camera.
- A Phidget Spatial Precision 3/3/3 IMU.
- An HP Z2 Mini G9 workstation featuring an Intel Core i9-13900K processor, 32 GB of RAM, and an NVIDIA RTX A2000 GPU with 12 GB of VRAM.
- A differential-drive mobile base equipped with two motorized drive wheels, a suspension system, and four passive caster wheels for stability and support.

In the real-world setting, the robot relies on the ZED 2i RGB-D camera for person and bottle detection, and on the IMU to register the acceleration peak produced when the bottle falls off the tray. The simulated robot model in Webots closely replicates the physical sensor configuration and kinematic properties of the real platform, including an equivalent IMU model and a simulated RGB-D camera with comparable field of view and resolution, allowing for a consistent comparison between simulated and real IMU measurements.

7.2 Experimental Environments

The experimental scenario is implemented both in a simulated environment and on a physical mobile assistant robot. In the simulated case, Webots is used, consistent with the platform used for the real robot. In an initial set of trials, the robot operates without perception, obtaining the object's position directly from the simulator. In a subsequent set of trials, a perception pipeline is developed and validated using the simulated camera to obtain information about the bottle and the person, before being deployed directly on the real robot to test the transferability between simulation and reality.

In the simulated version, the person remains stationary at a distance of approximately 5 m from the robot's initial position, so that the robot's task reduces to approaching a fixed target. In the real-world version, the person starts close to the robot and moves throughout the experiment, requiring continuous tracking.

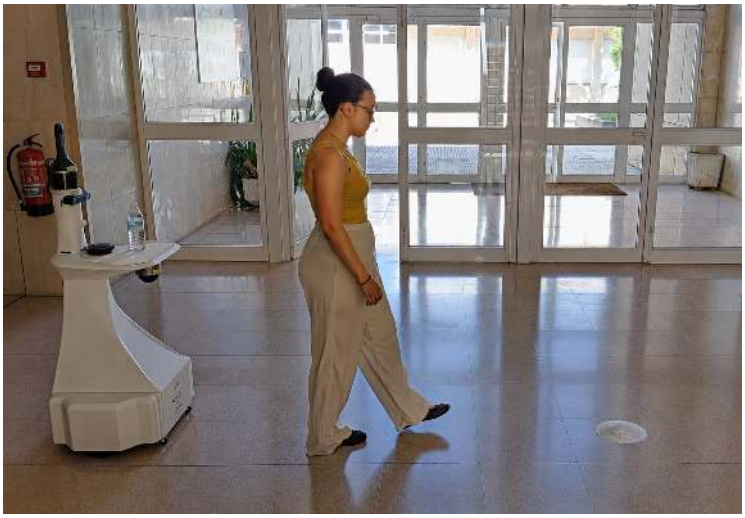


Fig. 4. The Shadow robot in the experimental scenario.

7.3 Task Description

The experiment consists of the robot following a person positioned in front of it while carrying a bottle on its tray. The bump is placed approximately 3 m from the robot's initial position, along its path toward the person. The same obstacle was used to induce the bottle's fall in both cases: a bump measuring 25 cm in width and length and 3.5 cm in height, modeled in 3D for use in Webots, which was subsequently 3D-printed and affixed to the floor in the real-world setting to preserve conditions similar to those of the simulated environment (see Fig. 5).

Fig. 5. The bump across the three spaces: (left) the 3D model, (center) the bump in Webots, and (right) the physical bump in the real-world environment.

As the robot follows the person, it simultaneously performs an internal simulation of its current belief about the task—that is, following a person across obstacle-free terrain while carrying a bottle. This belief simulation is executed by the Inner Simulator, using PyBullet as its underlying physics engine. Once the bottle falls and the mission is halted, the Semantic Memory Agent generates a set of candidate hypotheses, as described in Section 6.4, among which the

external hypothesis of an undetected terrain perturbation is considered. This hypothesis is accompanied by a simulation blueprint that specifies a grid of candidate bump positions and dimensions, consistent with the grounding anchors and expected observations associated with the hypothesis. The Inner Simulator then performs a second round of simulations, instantiating each candidate bump from the blueprint and reproducing the robot's trajectory over it. The resulting synthetic IMU measurements are compared against the real IMU measurements recorded by the Episodic Memory Agent at the moment of the fall, allowing the system to identify the bump configuration that best reproduces the observed acceleration peak.

Once the best-matching candidate is identified, a new node representing the bump is inserted into the DSR, connected to the robot through an RT edge that encodes its position relative to the robot (see Fig.6).

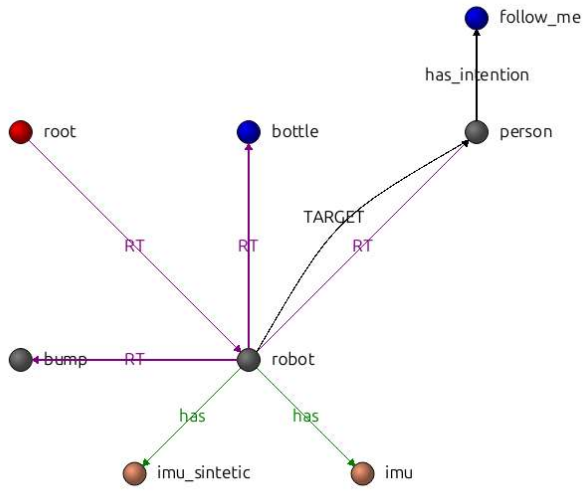


Fig. 6. Final state of the graph with the bump node.

8 Results and Discussion

The integration of both Semantic and Episodic Memory Agents enabled CORTEX to transition from basic anomaly detection to explicit causal explanation. Upon deletion of the bottle's RT edge, the Semantic Memory Agent intercepted the structural mismatch but was unable to formulate an immediate symbolic explanation, as the causal properties of a 'bump' were not yet integrated into its operational ontology. Consequently, the Semantic Memory Agent escalated the anomaly to the LLM-based hypothesis generation stage.

8.1 Candidate Generation

Because the generative stage is decoupled from any specific backend, evaluating it with a single model would conflate the behaviour of the method with the idiosyncrasies of that model [21]. We test the generator with three different Ollama Cloud models, spanning distinct families and parameter scales: `gpt-oss:120b`, `qwen3.5:397b` and `gemma4:31b`. Every other element of the pipeline was held fixed, leaving the model as the only independent variable. This isolates two properties that a single run cannot establish: whether the structured output schema holds true independently of the backend, and whether the causally relevant causes are consistently reflected, rather than being an artefact of a single model.

The three models returned well-formed batches of three internal and three external hypotheses, all compliant with the output schema (Table 1). The internal hypotheses stayed confined to subsystems actually declared in the robot self-model, naming the suspension, the wheels, the IMU, the on-board compute or the LiDAR explicitly, and no candidate invoked hardware the robot does not have.

Table 1. Hypotheses produced by three Ollama Cloud models for the same bottle-loss trigger. The mark † denotes the external hypotheses consistent with the ground-truth cause, an undetected terrain/obstacle perturbation.

Model	Internal hypotheses	External hypotheses
`gpt-oss:120b`	Suspension collapse IMU pos drift Auto-release during follow	Human pickup Mobile-platform collision Floor slope†
`qwen3.5:397b`	IMU drift, holonomic motion Suspension oscillation Perception latency	Human retrieval Unmapped-furniture collision† Ambient lighting
`gemma4:31b`	Suspension vibration slip Wheel lateral jerk Bpearl LiDAR interference	Human intervention Environmental obstacle collision† Air draft

Compared against the ground truth, an undetected terrain irregularity that destabilises the platform, two observations stand out. First, every model includes within its external family a local environmental perturbation consistent with the true cause (a floor slope, an obstacle or a furniture collision), so that the relevant cause direction is recovered by all three models even if the exact instantiation differs. Second, the models also diverge in their remaining proposals: alongside the terrain-related hypothesis, each suggests external causes (a human pickup, ambient lighting or an air draft) that are plausible in the abstract but cannot be checked against the robot's recorded experience. The generator therefore delimits a bounded space of plausible causes without determining, by itself, which one

holds; that decision is delegated to the physical, experience-grounded validation described next, in which each simulable candidate is instantiated and tested against the IMU trace recorded at the moment of the fall.

8.2 Candidate Simulation

The computational cost of the validation stage depends on the nature of the candidate hypothesis and, in particular, on the number of simulation iterations requested by the LLM-generated blueprint. Candidate hypotheses are simulated in parallel, so the overall duration of this phase is determined by the slowest candidate rather than by the sum of individual runs. Internal hypotheses, such as a transient fault in the drive motor, are instantiated as a single trajectory replay and complete in approximately 30 seconds. External hypotheses involving spatial uncertainty, such as the terrain-perturbation candidate, require sampling a grid of bump positions and dimensions over a 2-meter search area, which extends the simulation time to approximately one minute.

Once all simulations have finished, a second, sequential phase retrieves the corresponding measurements from the episodic memory and compares them by similarity against each simulated candidate. This processing phase takes approximately 20 seconds regardless of the hypothesis type, since the comparison cost depends on the length of the recorded trace rather than on the complexity of the simulated scenario. Table 2 summarises the timings for both phases.

Table 2. Approximate execution time of the candidate simulation (run in parallel) and the episodic data post-processing (run sequentially) for each hypothesis type.

Hypothesis type	Simulation time (s)	Post-processing time (s)
Internal (single trajectory)	∼30	∼20
External (2 m grid search)	∼60	∼20

8.3 Candidate Validation

Once the simulations were completed, the Semantic Memory Agent received the resulting execution data to evaluate the structural plausibility of the candidate cause. Within this verification, the system ensured that the newly instantiated 'bump' object conformed strictly to the core knowledge base of intuitive physics. The geometry and dimensions of the virtual protrusion were constrained to logical operational scales, ensuring the simulator discarded scenarios with anomalous physics or unrealistic structural proportions. The Semantic Memory Agent formally accepted the collision hypothesis only when the simulated run achieved two validation criteria: it maintained strict adherence to known physical laws, and its IMU simulated measurements yielded an optimal similarity against the real IMU and structural trajectories recorded.

8.4 Discussion

Compared to traditional reactive systems, which typically execute exception recoveries or freeze upon losing target tracking, our methodology allows the robot to contextualize environmental failures. Furthermore, data-driven approaches require massive training datasets of accidents to recognize a fall; conversely, CORTEX achieves data efficiency by solving the anomaly through a single real episodic experience augmented by internal simulations.

9 Conclusion and Future Work

The experiments reported here show that the joint operation of the semantic and episodic memories allows CORTEX to move beyond simple anomaly detection towards a grounded causal explanation of perceptual failures. The bottle-fall scenario was correctly attributed to an undetected terrain perturbation, and this explanation was recovered consistently across three different LLM backends, confirming that the relevant causal direction is not an artefact of a single model but a property of the proposed pipeline. By converting a physical failure into structured conceptual knowledge through a single real episodic experience augmented by internal simulations, rather than relying on massive accident datasets, the system is able to synthesise new perceptual concepts on the fly and extend its control architecture to anticipate similar hazards in the future. These outcomes demonstrate a practical pipeline for runtime anomaly diagnosis, bridging the gap between symbolic execution and generative hypotheses. Future work will focus on extending the range of detectable anomalies beyond the single-object case studied here, improving the efficiency and scalability of the candidate-cause search, and incorporating richer forms of human-robot interaction to support and refine the generated explanations.

Acknowledgments. This work has been supported by grants PID2022-137344OB-C31, PID2022-137344OB-C32 and PID2022-137344OB-C33, funded by MICIU/AEI/10.13039/501100011033 and by "ERDF/EU". Also by the FEDER Project 0124_EUROAGE_MAS_4_E under POCTEP Programme 2021–2027, and by the R&D&i project PID2022-137344OB-C31, supported by MICIU/AEI/10.13039/501100011033 and "FEDER, Una manera de hacer Europa". Additional co-funding was provided by the European Union through the European Regional Development Fund (85%) and by the Junta de Extremadura. The managing authority is the Ministerio de Hacienda (Spain).

References

1. Bavle, H., Sanchez-Lopez, J.L., Shaheer, M., Civera, J., Voos, H.: S-Graphs 2.0 – A Hierarchical-Semantic Optimization and Loop Closure for SLAM (Oct 2025). https://doi.org/10.48550/arXiv.2502.18044

2. Beßler, D., Porzel, R., Pomarlan, M., Vyas, A., Höffner, S., Beetz, M., Malaka, R., Bateman, J.: Foundations of the socio-physical model of activities (SOMA) for autonomous robotic agents. Frontiers in Artificial Intelligence and Applications **344**, 159–174 (2021). https://doi.org/10.3233/faia210379
3. Bustos, P., Manso, L., Bandera, A., Bandera, J., García-Varea, I., Martínez-Gómez, J.: The CORTEX cognitive robotics architecture: Use cases. Cognitive Systems Research **55** (2019)
4. Bustos, P., Bachiller, P., García-Varea, I., Martínez-Gómez, J., Bandera, A., Marfil, R.: Insight: The quest for causal explanations. In: 23rd International Workshop of Physical Agents (WAF 2023). pp. –. WAF, Aranjuez, Madrid, Spain (2023), presented in WAF 2023
5. Bustos, P., Bachiller, P., García-Varea, I., Martínez-Gómez, J., Bandera, A., Marfil, R., Calderón, J., Ballesteros-Jerez, J., Galeas, J.: Towards causal reasoning in robotics via semantic and episodic memory integration. In: 25th International Workshop of Physical Agents (WAF 2025). pp. –. WAF, Cartagena, Spain (2025), presented in WAF 2025
6. Bustos García, P., Manso Argüelles, L.J., Bandera, A., Bandera, J.P., García-Varea, I., Martínez-Gómez, J.: CORTEX: A Novel Cognitive Architecture for Social Robots. In: Proceedings of the EUCognition Meeting on Cognitive Robot Architectures. Vienna, Austria (2016)
7. Bustos García, P., García, J.C., Cintas Peña, R., Martinena Guerrero, E., Bachiller Burgos, P., Núñez Trujillo, P., Bandera, A.: Dsrd: A proposal for a low-latency, distributed working memory for cortex. In: Advances in Physical Agents II. pp. 109–122. Springer International Publishing, Cham (2021)
8. Bärmann, L., DeChant, C., Plewnia, J., Peller-Konrad, F., Bauer, D., Asfour, T., Waibel, A.: Episodic memory verbalization using hierarchical representations of life-long robot experience. In: 2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids). pp. 783–790 (2025). https://doi.org/10.1109/Humanoids65713.2025.11203101
9. Galeas, J., Tudela, A., Óscar Pons, Bandera, J.P., Bandera, A., Bustos, P.: Crdt-based knowledge synchronisation in an internet of robotics things ecosystem for ambient assisted living. Computer Vision and Image Understanding **259**, 104437 (2025). https://doi.org/10.1016/j.cviu.2025.104437
10. Guo, X., Jiang, C., Kim, H.B., Han, Y., Sun, Y., Xiao, Y., Yang, J.: Chameleon: Control-indexed prospective memory for visuomotor manipulation (2026), `https://arxiv.org/abs/2603.24576`
11. IEEE Standard for Ontologies for Robotics and Automation: IEEE standard for ontologies for robotics and automation. IEEE Std 1872-2015 pp. 1–60 (2015). https://doi.org/10.1109/IEEESTD.2015.7084073
12. Kang, H., Voloshynovskiy, S., Thalmann, N.M.: Human-inspired context-selective multimodal memory for social robots (2026), `https://arxiv.org/abs/2604.12081`
13. Nunez, P., Manso, L.J., Bustos, P., Drews-Jr, P., Macharet, D.G.: A Proposal for the Design of a Semantic Social Path Planner using CORTEX **1**(1) (2016)
14. Núñez Trujillo, P., Manso Argüelles, L.J., Bustos García, P., Drews Jr., P., Macharet, D.G.: Towards a new Semantic Social Navigation Paradigm for Autonomous Robots using CORTEX. In: RO-MAN 2016) - BAILAR2016 Workshop (2016)
15. Olivares-Alarcos, A., Foix, S., Borgo, S., Guillem Alenyà: OCRA – an ontology for collaborative robotics and adaptation. Computers in Industry **138**, 103627 (2022). https://doi.org/10.1016/j.compind.2022.103627

16. Ollama: Ollama. https://ollama.com, accessed: 2026-06-24
17. Ontotext: GraphDB. https://www.ontotext.com/products/graphdb/, accessed: 2026-06-24
18. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press, Cambridge, UK, 2nd edn. (2009)
19. Pearl, J., Mackenzie, D.: The Book of Why: The New Science of Cause and Effect. Basic Books, New York (2018)
20. Schmid, L., Abate, M., Chang, Y., Carlone, L.: Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments (2024), https://arxiv.org/abs/2402.13817
21. Sun, M., Yin, Y., Xu, Z., Kolter, J.Z., Liu, Z.: Idiosyncrasies in large language models (2025). https://doi.org/10.48550/arXiv.2502.12150
22. Tenorth, M., Beetz, M.: KnowRob — A knowledge processing system for cognition-enabled robot control. The International Journal of Robotics Research **32**(5), 566–590 (2013)
23. Tulving, E.: Episodic and Semantic Memory, pp. 381–403. Academic Press, New York (1972)
24. Vice, J., Ruiz-Sanchez, N., Douglas, P.K., Sukthankar, G.: Visual episodic memory-based exploration (2024), https://arxiv.org/abs/2405.11298

Building Incremental Memories of Human Activities from Mobile Robot Observations

Alberto Pérez-Álvarez[1], Javier Ballesteros-Jerez[1], Jesus Martínez-Gómez[1], and Ismael García-Varea[1]

Computing Systems Department, Universidad de Castilla-La Mancha, Spain
Alberto.Perez25@alu.uclm.es, {javier.ballesteros, jesus.martinez, ismael.garcia}@uclm.es

Abstract. Mobile robots can collect spatially distributed observations of human activity, but individual images provide only fragmented evidence of what occurs across an environment. This paper presents a platform-independent architecture that incrementally transforms visual observations into a spatially and temporally grounded textual memory. An efficient local detector identifies the presence and posture of people, while a vision-language model generates richer descriptions when required. The resulting memory supports natural-language queries during and after the patrol, context-aware risk notifications and post-round summarisation. Its usefulness for retrieval-augmented question answering is evaluated with RAGAS through predefined factual questions and three experiments analysing a) detector-assisted perception, b) spatial sampling density, and c) the preliminary use of short video clips. Detector-assisted perception improves the quality of the retrieved evidence while avoiding unnecessary generative inference. Across five spatial sampling intervals ranging from 0.2 to 4 m, answer correctness remains around 0.6, with no monotonic degradation as the number of observations decreases. Configurations between 0.5 and 1 m obtain the strongest retrieval-oriented results, suggesting that moderate subsampling can remove redundant observations without substantially reducing the information available to the chatbot. Video clips achieve a correctness of 0.60 but produce weaker retrieval-oriented metrics and require further analysis. These preliminary findings support the use of spatially grounded image descriptions for incrementally building accessible memories of human activities and motivate their future validation in real assisted-living environments.

Keywords: Mobile Robots · Vision-Language Models · Retrieval-Augmented Generation · Human Activity Monitoring · Assisted Living

1 Introduction

Mobile robots can act as active sensing platforms in residences, hospitals and other assisted-living environments [4]. Unlike fixed cameras, they can visit different areas, associate observations with their estimated location and collect spatially distributed information about occupancy and human activity. This information can support care staff through risk notifications and descriptions of

relevant events. However, a single static summary may not satisfy the information needs of different users. Staff may instead need to ask specific questions about a room, a person or a recent activity. A conversational interface makes this information accessible through natural language and does not require users to have technical knowledge of the underlying perception or retrieval system. These applications must nevertheless consider privacy, human supervision and the operational requirements of care facilities [13].

Vision-language models (VLMs) can transform visual observations into flexible natural-language descriptions [8]. Applying them indiscriminately to every image captured during a patrol may nevertheless introduce latency, computational demand or external service costs. Efficient computer vision models provide a complementary source of information. Person detectors and pose estimators can run locally, identify whether human activity is present and provide structured evidence such as person counts and body keypoints. Their outputs are less expressive than VLM descriptions, but they are well suited to filtering observations and supplying additional perceptual context [14].

The proposed system combines an efficient human detector with a VLM. YOLO is used in the current implementation, although the architecture is not tied to this particular model. When no person is detected, the VLM is skipped and the system records that no human activity has been observed at the current location. When people are present, the image and the structured information produced by the detector are provided to the VLM. This combination aims to preserve the efficiency and reliability of specialised vision methods while benefiting from the descriptive flexibility of multimodal generative models.

The generated descriptions are associated with spatial and temporal metadata and incorporated incrementally into a textual observation memory. This memory constitutes the retrieval corpus of a retrieval-augmented generation system [7]. As the robot moves through the environment, new observations become immediately available to a chatbot that can answer questions during or after the patrol. The system therefore provides interactive access to the evolving state of the residence rather than limiting users to a predefined final report. The same observations can also support immediate risk notifications and the generation of a global summary once the patrol has finished.

The proposed approach is evaluated with RAGAS [2] through three experiments. The first experiment examines whether combining a specialised human detector with a VLM improves the information stored in the observation memory and the answers generated from it. The second experiment studies the effect of reducing the spatial sampling density, with the aim of identifying how many visual observations are needed to represent the environment without introducing excessive redundancy. The third experiment provides a preliminary analysis of the potential of video-capable multimodal models by comparing individual images with short video clips. This experiment explores whether temporal information can improve the representation of human activities or whether clip-level descriptions reduce the precision of the records used by the retrieval system. The

architecture is independent of a particular robotic platform, and TIAGo is used only to conduct the experiments in a simulated hospital environment.

2 Related Work

Mobile robots have been investigated as active sensing platforms in assisted-living environments, where their mobility allows several areas to be monitored using a single system [4]. Human-activity analysis has traditionally relied on person detection, pose estimation and action-recognition methods [15]. These approaches provide efficient and structured outputs but are usually limited to predefined categories. Vision-language models offer a complementary capability by generating open-ended descriptions of people, activities and scene context [8]. Video-capable models can also exploit temporal information, although aggregating several moments into one description may reduce the granularity of the evidence available for later retrieval [9].

Visual streams often contain considerable redundant information, motivating the selection of relevant observations before VLM inference. Previous work [1] compared equidistant, appearance-based and semantic keyframe-selection strategies, showing that reducing the number of processed frames can preserve descriptive information and limit the generation of irrelevant actions. The present work follows this direction in the context of mobile-robot patrols. It combines an efficient local detector with a VLM and studies how the spatial sampling interval affects the information retained about the environment.

Retrieval-augmented generation extends the capabilities of language models by conditioning their responses on evidence retrieved from an external and updateable knowledge source [7]. Unlike the fixed parametric knowledge encoded in the model, the retrieved corpus can incorporate new domain information without retraining. In this work, the corpus is an incrementally constructed textual observation memory containing spatially and temporally grounded descriptions of the robot's observations. FAISS supports similarity-based retrieval from this memory [5], while RAGAS evaluates the generated answers and their relationship with both the reference answers and the retrieved evidence [2].

3 Proposed Architecture

The proposed architecture incrementally transforms visual observations collected during a robot patrol into spatially and temporally grounded textual records. As shown in Fig. 1, the robot captures images or short video clips according to a configurable spatial sampling policy and associates them with the corresponding area of the environment. An efficient local perception model first extracts structured information about the presence and posture of people, while a vision-language model provides richer scene descriptions when required. The resulting records are incorporated into an incremental observation memory that supports natural-language dialogue, context-aware risk notifications and post-round summarisation.

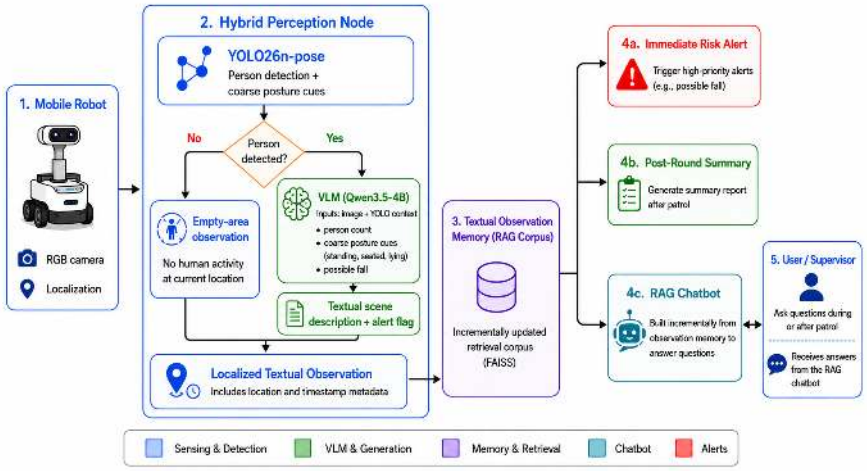


Fig. 1. Proposed architecture. An efficient local perception model extracts structured information and determines whether generative visual inference is required. The resulting spatially grounded observations support incremental question answering, context-aware notifications and post-round summarisation.

3.1 Mobile sensing and semantic spatial grounding

The architecture requires a mobile platform equipped with an RGB camera and a localisation system. During a patrol, the localisation module provides the robot pose, while a semantic layer associates that pose with a meaningful area of the environment, such as a patient room, corridor or reception area. Semantic maps extend geometric representations with conceptual information that can support robot reasoning and task execution [3]. In the proposed system, this information allows every visual observation to be grounded in a named area and subsequently interpreted according to its spatial context.

The experimental implementation uses ROS 2 and Gazebo [10, 6]. A simulated TIAGo robot provides the mobile base and RGB camera, while the AWS RoboMaker Hospital World [1] represents the care environment. The environment contains 20 configurable zones, together with a reception area and connecting corridors. Simulated humans perform activities such as standing, sitting, lying down, walking, running and moving another person in a wheelchair. SLAM Toolbox is used to generate the occupancy map [11], and Nav2 controls navigation between predefined areas [12]. Figure 2 shows examples of the simulated environment, the represented human activities and the structured pose information generated by the local perception model.

3.2 Hybrid local and generative perception

The camera captures RGB images at a resolution of 640×480 pixels. Each observation is first processed by an efficient person and pose detector that can

[1] https://github.com/aws-robotics/aws-robomaker-hospital-world

(a) Empty common area (b) Simulated human activity (c) Structured pose output

Fig. 2. Examples of the simulated care environment, human activity and structured pose information.

run locally on the robotic platform. The architecture is independent of a particular detector, provided that it can identify people and return suitable structured information. In the experimental implementation, this role is performed by `yolo26n-pose`[2]. The detector estimates body keypoints for each identified person. Geometric rules applied to these keypoints provide coarse posture cues, including standing, sitting, lying down and possible falls.

When no person is detected, the generative model is not invoked. Instead, the system creates a textual record indicating that no human activity has been observed in the corresponding semantic area. When one or more people are present, the visual observation is processed by the Qwen3.5-4B vision-language model [3]. The prompt additionally includes the number of detected people and the available posture cues. The generative model therefore receives both the original visual input and structured evidence produced by the local perception stage. The final record also contains the observation location, timestamp and available detector metadata. This common format allows records generated from images or video clips, and with or without generative inference, to be processed by the same downstream components.

3.3 Incremental memory, dialogue and context-aware services

Each generated description is converted into a LangChain `Document` and inserted into a FAISS-backed textual observation memory [5]. One document is generated for each image or video clip, without additional text splitting. Its content includes the scene description and the associated spatial and temporal metadata. Since documents are inserted as soon as they are generated, the memory evolves throughout the patrol and provides an up-to-date representation of the observed environment.

The RAG pipeline combines semantic and lexical retrieval, followed by a reranking stage. A Streamlit chatbot allows non-technical users to ask natural-language questions about the residence while the patrol is in progress or after it has finished. The answer-generation model uses temperature 0 to reduce output variability. The interface employed to access the incremental observation memory is shown in Fig. 3.

[2] https://docs.ultralytics.com/models/yolo26/
[3] https://huggingface.co/Qwen/Qwen3.5-4B

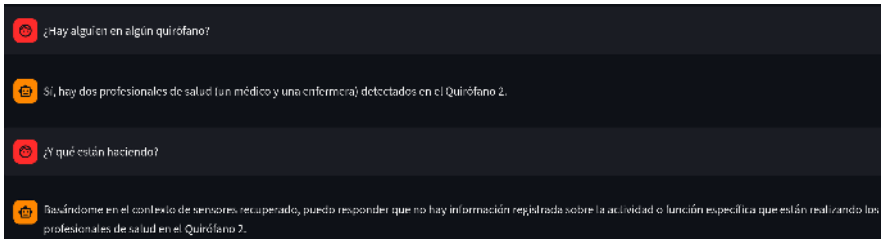


Fig. 3. Conversational interface for querying the incremental observation memory during or after a robot patrol.

The generated observations can also support immediate risk notifications. However, a posture estimate should be treated as perceptual evidence rather than as an alarm condition by itself. The interpretation of a lying posture depends not only on the semantic area but also on the surrounding objects and their spatial relationship with the person. For example, a person lying on a bed in a patient room may correspond to normal activity, whereas a person lying on the floor and spatially separated from the available beds may indicate a possible fall. The notification mechanism can therefore combine three complementary sources of information: posture cues obtained from the local perception model, scene elements and spatial relationships described by the VLM, and the semantic label associated with the robot location. This contextual combination avoids relying on simple rules such as associating every lying posture with an emergency or assuming that such a posture is always normal inside a patient room.

Figure 4 illustrates this situation. Although the semantic map identifies the area as a hospital room, the local detector identifies a person in a lying posture and the visual description places that person on the floor, away from the beds. The joint interpretation of these elements causes the observation to be flagged as a potentially dangerous situation requiring human review.

Fig. 4. Example of context-aware risk notification in a patient room. The local perception model identifies a person in a lying posture, while the scene context indicates that the person is on the floor and spatially separated from the beds. Their combination causes the situation to be flagged for human review.

The current alerting mechanism should be understood as a preliminary decision-support component rather than as a clinically validated fall-detection system. The system reports a potentially dangerous situation, but a human supervisor remains responsible for interpreting the available evidence and initiating any appropriate response.

Finally, the incremental observation memory can be used to generate a global summary after the patrol. This summary is intended to provide an overview of the state of the residence, including relevant activities, occupied areas and potentially dangerous situations. Its quality is not evaluated in the present work. Future experiments will analyse how the summary can be shortened while preserving relevant information, with particular attention to safety-critical events.

4 Experimental Methodology

The experiments were conducted in the simulated hospital described in Section 3. All perception, memory construction, retrieval and answer-generation components were executed on the same personal computer equipped with a dedicated GPU. The evaluation is based on 21 manually defined factual questions concerning occupancy, person counts and activities observed in particular areas of the environment. Representative questions ask how many people were observed in the reception area, whether a particular room was empty, or which activities occurred in a corridor. Each question is associated with a reference answer derived from the simulated scenario.

All evaluated configurations were obtained from a single robot patrol to preserve the activities, viewpoints and environmental conditions represented in the observations. The original sequence contains 1,273 spatially grounded images captured at intervals of approximately 0.2 m. Four additional image sequences were generated by spatially subsampling this sequence at intervals of approximately 0.5, 1, 2 and 4 m. Consequently, the configurations differ in observation density while retaining observations from the same patrol. For the exploratory video experiment, selected consecutive observations were grouped into 182 clips containing five frames each, with clips sampled at intervals of approximately 2 m. Table 1 summarises the resulting visual representations.

Table 1. Visual representations derived from the same robot patrol.

Representation	**Spacing**	**Items**	**Frames/item**	**Size**
S-0.2m (non-reduced sequence)	0.2 m	1,273	1	55.7 MB
S-0.5m (small reduction)	0.5 m	465	1	19.2 MB
S-1.0m (medium reduction)	1.0 m	264	1	10.8 MB
S-2.0m (hard reduction)	2.0 m	160	1	6.5 MB
S-4.0m (extreme reduction)	4.0 m	102	1	3.9 MB
Video clips	2.0 m	182	5	139 MB

Three experiments were conducted. The first compares two perception strategies using the complete sequence of 1,273 images. In the baseline configuration, every image is processed directly by the VLM. In the detector-assisted configuration, the local perception model first determines whether people are present. Observations without detected people are converted directly into textual records, avoiding VLM inference. When people are detected, the image is processed by the VLM together with the detected person count and the available posture cues.

This experiment therefore evaluates the combined effect of detector-based filtering and contextual enrichment; their individual contributions are not analysed separately. The second experiment evaluates the detector-assisted pipeline using the five spatial sampling intervals shown in Table 1. Since all reduced sequences are obtained from the same patrol, this comparison analyses the effect of removing spatially redundant observations without introducing changes in the represented activities or environmental conditions. The third experiment provides a preliminary comparison between image-based and clip-based observation memories. The video configuration groups consecutive images into five-frame clips and is compared primarily with S-2.0m, since both representations use an approximate spatial interval of 2 m. Nevertheless, the different number and temporal extent of their retrieval units mean that this experiment should be interpreted as exploratory rather than as a fully controlled comparison.

Question-answering performance is evaluated using five RAGAS metrics: answer correctness, answer relevancy, faithfulness, context precision and context recall [2]. Answer correctness measures agreement with the manually defined reference answer, while answer relevancy evaluates how directly the generated response addresses the question. Faithfulness measures whether the statements in the generated answer are supported by the retrieved observations and must therefore not be interpreted as direct agreement with the simulated ground truth. Context precision evaluates the relevance of the retrieved observations, whereas context recall measures whether they cover the information required by the reference answer. The reported scores are aggregated over the same 21 questions for every configuration.

5 Experiments and Results

Three experiments analyse how the proposed architecture is affected by the perception strategy, the spatial density of the observations and the visual input modality. All evaluated configurations are derived from the same robot patrol, as described in Section 4, and all RAGAS results correspond to the 21 factual questions. Table 2 summarises the results.

Table 2. Results (best in bold) for the experiments. C: answer correctness; R: answer relevancy; F: faithfulness; CP: context precision; CR: context recall.

Configuration	C	R	F	CP	CR
VLM, original images	0.59	0.34	0.71	0.22	0.62
S-0.2m (detector-assisted)	0.60	0.55	0.83	0.82	0.88
S-0.5m (detector-assisted)	0.59	0.50	0.82	**0.93**	**0.91**
S-1.0m (detector-assisted)	0.62	0.52	**0.93**	0.85	**0.91**
S-2.0m (detector-assisted)	0.57	**0.58**	0.86	0.73	0.81
S-4.0m (detector-assisted)	**0.64**	0.55	0.78	0.82	0.87
Detector-assisted, video clips	0.60	0.13	0.25	0.00	0.41

5.1 Experiment 1: Contribution of detector-assisted perception

The first experiment studies whether the structured information produced by an efficient local detector contributes to the construction of a more useful textual observation memory while reducing the computational cost of processing a complete patrol. Both configurations use the original sequence of 1,273 images. In the first configuration, every image is processed directly by the VLM. In the second, the local detector determines whether human activity is present and provides person counts and posture cues when people are detected. YOLO is used for this purpose in the experimental implementation, although the comparison concerns the general detector-assisted perception strategy rather than this particular model.

The detector-assisted configuration increases all the metrics, including the most relevant ones: answer correctness from 0.59 to 0.60 and answer relevancy from 0.34 to 0.55. Context precision rises from 0.22 to 0.82, while context recall increases from 0.62 to 0.88. These results indicate that the resulting memory contains evidence that is both more relevant to the questions and more effective at covering the information required by the reference answers. Faithfulness also increases from 0.71 to 0.83. This metric evaluates whether the statements contained in an answer are explicitly supported by the retrieved context, rather than whether the answer agrees with the manually defined reference. All the improvements support the use of specialised local perception as a complementary source of evidence for the VLM.

The detector-assisted strategy also substantially reduces the time required to process the complete image sequence. As shown in Table 3, the mean processing time per observation decreases from 14.079 to 3.207 s, corresponding to a reduction of 77.2% and a speed-up of approximately 4.4×. The median decreases to 0.047 s because most observations contain no detected people and can therefore be converted into textual records without invoking the VLM. The local detector requires only 0.046 s on average, whereas VLM inference accounts for 98.53% of the processing time in the detector-assisted configuration.

Table 3. Processing time (s) per observation over the original sequence of 1,273 images.

Configuration	Mean	Median
VLM only	14.079	13.235
Detector-assisted	**3.207**	**0.047**

5.2 Experiment 2: Influence of the spatial sampling interval

The second experiment analyses how the spatial sampling interval affects the information preserved in the textual observation memory. All configurations are derived from the same robot patrol and use the same detector-assisted perception pipeline. The original sequence, denoted as S-0.2m, contains observations captured approximately every 0.2 m. Four spatially subsampled variants were

generated from it: S-0.5m, S-1m, S-2m and S-4m. This design allows the effect of progressively reducing the number of observations to be studied while preserving the activities and environmental conditions represented in the original patrol. Table 2 shows that the relationship between sampling density and question-answering performance is not monotonic. Answer correctness varies within a relatively narrow range, from 0.57 to 0.64, and the highest value is obtained by S-4m. S-0.5m and S-1m achieve the highest context recall while S-0.5m also obtains the best context precision. In contrast, S-2m produces the lowest correctness, context precision and context recall, despite achieving the highest answer relevancy.

These results indicate that reducing the number of observations does not necessarily lead to a proportional loss of information. Moderate subsampling may remove visually redundant records and produce a more focused retrieval corpus. However, the strong variations observed in some metrics and the good correctness obtained with the sparsest configuration require cautious interpretation. With only 21 factual questions, individual successes or failures can have a noticeable effect on the aggregate scores. The results therefore suggest that sampling intervals between 0.5 and 1 m offer a favourable balance between observation density and retrieval quality, but additional sequences and a larger question set will be required to determine whether this behaviour generalises.

5.3 Experiment 3: Preliminary analysis of video clips

The third experiment provides a preliminary analysis of the potential of video-capable multimodal models for constructing the textual observation memory. Consecutive observations from the patrol were grouped into 182 short clips containing five frames each and compared with the reduced image sequence. The video-based configuration achieves an answer correctness of 0.60, close to the values obtained with individual images. However, context recall decreases to 0.41, while context precision reaches 0.00. These results suggest that short clips can preserve part of the information required to answer factual questions, but their broader descriptions may constitute less precise retrieval units than individual spatially grounded images. Some of the observed metric variations, particularly the absence of context precision, require further analysis to determine whether they arise from the clip descriptions, the retrieval process or the evaluation setup. A more complete evaluation will require controlled acquisition conditions, video-specific prompts and retrieval mechanisms designed for temporally extended observations.

6 Conclusions and Future Work

This paper has presented a platform-independent architecture for incrementally transforming visual observations collected by a mobile robot into spatially and temporally grounded textual records. By combining efficient local perception with a vision-language model, the system constructs an evolving observation

memory that supports natural-language dialogue, context-aware risk notifications and post-round summarisation. The preliminary results obtained in simulation suggest that the proposed approach has the potential to be transferred to real assisted-living environments.

The detector-assisted perception strategy reduces the mean processing time per observation by 77.2% while improving several retrieval-oriented metrics. The spatial sampling experiment further shows that reducing the observation density does not produce a monotonic degradation in performance. Across intervals ranging from 0.2 to 4 m, answer correctness remains between 0.57 and 0.64, while the configurations sampled every 0.5 and 1 m obtain the strongest context precision and recall results. This behaviour suggests that moderate subsampling can remove redundant observations and produce a more focused retrieval corpus. However, the variations across metrics and the limited number of questions require cautious interpretation. The exploratory video experiment achieves comparable correctness but weaker retrieval-oriented results, whose causes require further analysis.

Future work will extend the evaluation with longer sequences, repeated patrols, additional environments and a larger set of questions. Greater variability in human activities, viewpoints and occupancy patterns will be introduced to determine whether the observed sampling behaviour generalises. Detector-based filtering and contextual enrichment will also be analysed separately, while image and video configurations will be compared under equivalent acquisition conditions and with retrieval strategies adapted to temporally extended observations.

The main next step will be to deploy the architecture on a physical mobile robot in a real residential or assisted-living environment. This deployment will allow the system to be evaluated under localisation uncertainty, illumination changes, occlusions and interactions with residents and staff. Particular attention will be paid to assessing the chatbot with non-technical users in terms of correctness, usefulness, ease of interaction and perceived trustworthiness, together with the complementary value of post-round summaries and context-aware notifications.

Acknowledgments. This work has been supported by grant PID2022-137344OB-C33, funded by MICIU/AEI/10.13039/501100011033 and by “ERDF/EU”. It has also been supported by project "Innovación Tecnológica y Social en Materia de Robótica" funded by the European Union NextGenerationEU/PRTR within the framework of "Convenio de Colaboración entre la Consejería de Bienestar Social de la JCCM y la UCLM".

References

1. Casas, A., Martínez-Gómez, J., de la Ossa, L.: Efficient real-time scene description with vision-language models. In: Proceedings of the 25th International Workshop on Physical Agents. pp. 119–131. Universidad Politécnica de Cartagena (2025)

2. Es, S., James, J., Anke, L.E., Schockaert, S.: RAGAS: Automated evaluation of retrieval augmented generation. In: Proceedings of the 18th conference of the european chapter of the association for computational linguistics: system demonstrations. pp. 150–158 (2024)
3. Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernández-Madrigal, J.A., González, J.: Multi-hierarchical semantic maps for mobile robotics. In: Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2278–2283 (2005)
4. Gomez-Donoso, F., Escalona, F., Rivas, F.M., Cañas, J.M., Cazorla, M.: Enhancing the ambient assisted living capabilities with a mobile robot. Computational intelligence and neuroscience **2019**(1), 9412384 (2019)
5. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data **7**(3), 535–547 (2019)
6. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2149–2154 (2004)
7. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems. vol. 33, pp. 9459–9474 (2020)
8. Li, J., Li, D., Savarese, S., Hoi, S.C.H.: BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 19730–19742 (2023)
9. Li, K., He, Y., Wang, Y., Li, Y., Wang, W., Luo, P., Wang, Y., Wang, L., Qiao, Y.: VideoChat: Chat-centric video understanding. Science China Information Sciences **68**, 200102 (2025)
10. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics **7**(66), eabm6074 (2022). https://doi.org/10.1126/scirobotics.abm6074
11. Macenski, S., Jambrecic, I.: SLAM toolbox: SLAM for the dynamic world. Journal of Open Source Software **6**(61), 2783 (2021)
12. Macenski, S., Martín, F., White, R., Ginés Clavero, J.: The marathon 2: A navigation system. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2718–2725. IEEE (2020)
13. Ravi, S., Climent-Pérez, P., Florez-Revuelta, F.: A review on visual privacy preservation techniques for active and assisted living. Multimedia Tools and Applications (2024)
14. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788 (2016)
15. Zakka, V.G., Dai, Z., Manso, L.J.: Action recognition in real-world ambient assisted living environment. Big Data Mining and Analytics **8**(4), 914–932 (2025)

Actionable Scene Graphs

Noé Zapata[1], Pedro Núñez[1], and Pablo Bustos[1]

RoboLab, Universidad de Extremadura, Spain
pbustos@unex.es
http://robolab.unex.es

Abstract. Scene graphs have become a dominant paradigm for explicitly representing objects and their relationships; however, they are usually constructed passively on the basis of teleoperational movements or exploration policies, meaning that the representation plays no role in the robot's actions. We propose Active Actionable Scene Graphs (ASG), built around a single principle: one free-energy functional minimises both perception and action. Each concept node of the CORTEX working memory (room, table, chair,...) is a probabilistic generative model refined by minimising a variational free energy (VFE) over differentiable signed distance fields; its residual uncertainty becomes an *epistemic affordance*, and a controller selects the trajectory that minimises the expected free energy (EFE). As both optimise the same quantity, the graph stops being a map and becomes a controller: the robot "moves to represent". We report preliminary results on the Shadow robot in Webots that illustrate this unified perception–action loop, with perception and action driven by the same free-energy minimisation.

Keywords: Active Inference · Scene Graphs · Free Energy · Active Perception · Cognitive Architectures

1 Introduction

Scene graphs have emerged as a dominant paradigm for explicitly representing objects and their relationships [1, 3, 9, 10, 14, 15]. In almost all of these systems, however, the graph is *descriptive* rather than *actionable*: it is built passively, while an externally supplied trajectory drives the robot, and the representation plays no part in deciding where to look next. This decouples mapping from acting, at odds with how biological agents acquire spatial knowledge, where perception and movement form a single closed loop.

We propose Active Actionable Scene Graphs (ASG), where the scene graph is a core component of the CORTEX architecture's working memory [5, 6]. Cognitive architectures like CORTEX excel at modular engineering (Fig. 1) but lack a theory that binds perception and action jointly; we supply one by reframing the architecture under Active Inference (AI), grounding both sensory assimilation and behavioural selection in the minimisation of a single variational free energy [8, 12]. Each concept node (room, table, chair) is a generative model whose *residual uncertainty is itself an action proposal*: acting to reduce it is what

drives the robot's next move. The graph thus stops being a map and becomes a controller, and the robot "moves to represent". This paper describes an ongoing project and reports initial results.

2 Related Work

Semantic mapping has moved from flat metric–semantic maps towards hierarchical, graph-structured representations. 3D scene graphs added a layered abstraction over objects, places and rooms [1], with real-time systems such as Kimera [14], scalable foundations [10], and open-vocabulary, functional and situational variants [3, 9, 15]. Across this literature the graph is built *passively*: the trajectory is supplied externally and the map is a by-product of perception.

Active Inference and the free-energy principle, in contrast, cast perception, learning and action as the minimisation of a single variational quantity [8], and a growing body of work applies them to robotics [4, 12], almost always at the level of low-level motor control. A complementary line—active perception and active SLAM—chooses motions that maximise expected information gain [2, 7], with recent work connecting expected-free-energy formulations to graph-based reasoning [16]. What is missing is a formulation that lifts this principle from the motor level to the object-centric level of the scene graph.

Classical cognitive architectures such as SOAR and ACT-R organise cognition around symbolic, centralised memories [11, 18]; CORTEX [5, 6] brings the idea to embodied robotics with a distributed, metric–symbolic working memory (Fig. 3). All remain engineering frameworks with no theory binding perception and action. We fill this gap by giving the CORTEX working memory a formal semantics: every node is a generative model under the free-energy principle, so one objective drives both a node's refinement (minimising its VFE) and the action it proposes to reduce that energy (a controller minimising the EFE)—a single loop at the level of the scene graph.

3 ASG: One Free-Energy Objective for Perception and Action

3.1 Working Memory as Coupled Generative Models

The backbone of CORTEX is the Deep State Representation (DSR), a directed graph $G = (V, E)$ whose nodes denote entities (robot, rooms, objects, intentions) and whose edges denote spatial transforms and semantic relations [5, 6]; it is called *deep* because of its hybrid geometric–symbolic nature. The DSR is the working memory and the only channel between agents: each agent keeps an editable local replica kept eventually consistent by a CRDT layer, so a perception agent can update a node at sensor rate while the controller reads its uncertainty to plan, both without blocking. A layer of RoboComp [13] components bridges the architecture to the world through ZeroC Ice [20] proxies, so the same code runs on the physical Shadow platform [17] or in Webots. In

an AASG, a concept node is not a deterministic record but a *probabilistic belief*: it stores not a point estimate of its entity but a posterior distribution over its parameters—a mean $\boldsymbol{\mu}$ and a covariance $\boldsymbol{\Sigma}$—while edges carry the transforms that map robot-frame evidence into each node's local frame. The scene graph is therefore a *probabilistic* representation, the shared state over which the free-energy minimisation operates; the open-source implementation is at `https://github.com/robocomp/active_inference`.

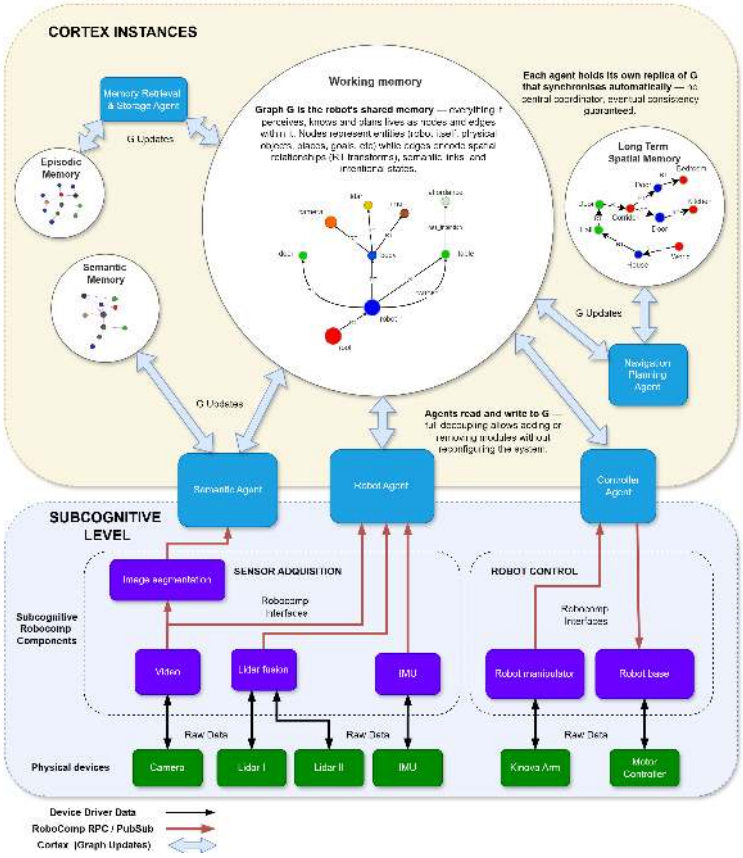


Fig. 1: The CORTEX architecture: specialised agents share the distributed DSR working memory G instead of communicating directly, with a RoboComp/Ice layer bridging sensors and actuators to the graph (Sect. 3).

3.2 Object Models based on Active Inference

The scene graph is populated by a set of CORTEX agents that detect, instantiate and maintain the concepts present in the environment. Each agent

(`room_concept`, `table_concept`, ...) owns one concept class: it segments the incoming evidence, instantiates a node in the working memory the first time the concept is observed, and from then on keeps that node's belief alive. Perception is framed as a variational inference problem in which every node maintains a probability distribution over the parameters $\boldsymbol{\theta}$ of its concept—a geometric and pose state whose dimension and meaning are fixed by the class. All classes share a common optimisation framework that minimises a sliding-window realisation of the Variational Free Energy (VFE),

$$\mathcal{F}(\boldsymbol{\theta}) = \mathcal{F}_{lik}(\boldsymbol{\theta}) + \mathcal{F}_{prior}(\boldsymbol{\theta}), \tag{1}$$

where i indexes the sensory points of the observed cloud. The likelihood-bound energy $\mathcal{F}_{lik} = \frac{1}{N}\sum_i \frac{w_i}{\sigma^2_{obs}}\rho(\mathrm{SDF}(\mathbf{p}_i;\boldsymbol{\theta}))$ scores each observation against a class-specific, differentiable Signed Distance Function that encodes the concept's expected geometry (ρ a robust loss limiting outliers, w_i a per-point weight), while $\mathcal{F}_{prior} = \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu})^\top \boldsymbol{\Lambda}(\boldsymbol{\theta}-\boldsymbol{\mu})$ is a Gaussian penalty anchoring the parameters to their prior and previous estimate, with a block-structured precision $\boldsymbol{\Lambda}$ so that size, position, orientation and the temporal transition are weighted independently. What changes between classes is only this parameter vector and its SDF: the `room` estimates a robot pose $q = [x, y, \theta]^\top$ against a fixed polygonal layout through a LiDAR-to-segment distance, whereas object classes compose simple differentiable primitives (slabs, boxes, cylinders) into a shape-and-pose SDF; the same energy and optimiser then apply unchanged. Because Eq. (1) is differentiable almost everywhere, the gradient $\partial\mathcal{F}/\partial\boldsymbol{\theta}$ is obtained exactly by automatic differentiation and the belief is refined in real time by gradient descent, replacing the explicit correspondence search of ICP-style alignment with a direct descent on a smooth energy.

3.3 Action: Epistemic Affordances

Action in the ASG is given an equal role to perception, closing the Active Inference loop. Each concept agent not only refines its own belief but also identifies and proposes an *epistemic affordance* (a targeted action designed to maximise information gain and further reduce the model uncertainty, i.e. to minimise the node's future VFE. For instance, `table` instances monitor face-coverage deficits across their local surfaces using the binned queue; if an under-observed face is detected, the agent computes an optimal viewpoint pose) a stand-off along the face's outward normal, oriented towards its centre, $\mathbf{v}^* = \mathbf{c}_{\text{face}} + d_{\text{view}}\,\hat{\mathbf{n}}_{\text{face}}$—and uses a Fisher information-gain proxy $\Delta H_f \propto A_{\text{face}}/\sigma^2$ to formalise the expected reduction in VFE. Crucially, the proposal is itself written into the working memory: the agent attaches an `affordance` node to its object node through a `has_intention` edge, advertising the target pose and its expected gain. The action request is therefore an inspectable part of the scene graph (Fig. 2), available to any other agent, rather than a hidden internal variable.

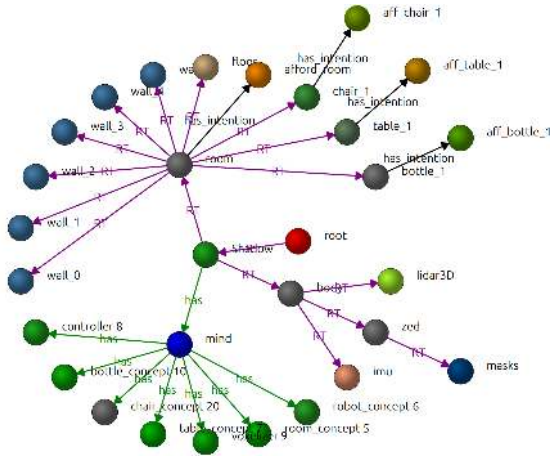


Fig. 2: Working-memory (DSR) probabilistic graph during a heterogeneous run: the `room` node with its walls and floor, the object concepts `table`, `chair` and `bottle`, each carrying a `has_intention` edge to its epistemic `affordance` node (`aff_*`), and the agent tier under `mind` (controller, concept agents, voxelizer) with the sensors under `body`; `RT` edges carry the spatial transforms. Each concept node holds a posterior belief over its pose and shape—a mean and covariance—rather than a fixed label, so the graph itself is probabilistic. Every affordance is thus an inspectable node in the shared graph.

3.4 EFE Evaluation and MPPI-based Control

The affordances proposed by the concept agents are gathered by a centralised `controller` agent that reads the `has_intention` edges and drives the robot in two stages.

Target selection chooses the affordance with the best Expected Free Energy (EFE), trading the predicted information gain of a viewpoint $\mathbf{v}$ against the cost of reaching it from the robot position $\mathbf{x}$,

$$G(\mathbf{v}) = \underbrace{\beta\,\|\mathbf{v}-\mathbf{x}\|}_{\text{pragmatic (travel)}} - \underbrace{\Delta\mathcal{H}(\mathbf{v})}_{\text{epistemic (info gain)}}, \tag{2}$$

where $\Delta\mathcal{H}$ is the predicted Fisher information gain of the targeted belief: a D-optimality gain on the robot pose—the Fisher matrix of the *visible* room corners, each contributing $J^\top J/\sigma^2$ through its observation Jacobian J, sharpening the precision $\boldsymbol{\Lambda} = \Sigma^{-1}$—or the coverage gain advertised by an object agent.

Trajectory generation then drives the robot to the selected target with a Model Predictive Path Integral (MPPI) planner [19]: it rolls out 100 candidate control sequences over a 50-step, $\Delta t = 0.1\,\mathrm{s}$ horizon under the robot's motion model, scores each by a cost $G(\pi)$ penalising distance and heading to the target, obstacle proximity and dynamic limits, and executes the weighted-average command $\sum_k w_k\,\pi_k$, with $w_k \propto \exp(-G(\pi_k)/\lambda)$, in receding-horizon

fashion. This softmax weighting is the Active-Inference posterior over policies $q(\pi) \propto \sigma(-\gamma\, G(\pi))$: the command is a Bayesian average of policies under their free energy rather than a hard minimum, and the temperature $\lambda = \gamma^{-1}$ is adapted online to keep the effective sample size healthy.

Because targets are chosen by predicted information gain rather than a hand-designed reward, the robot "moves to represent", proactively exploring to build its world model even before a specific human mission is assigned. Figure 3 shows the state of all the agents after driving the robot for some time.

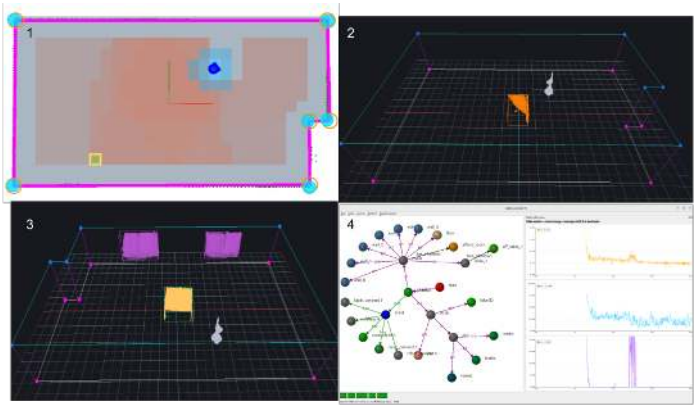


Fig. 3: Working state of the ASG agents driving the Shadow robot [17] in Webots. Top-left: the *room_ concept* agent fits the room layout and corners (shaded cells mark epistemic targets to visit). Top-right / bottom-left: the LiDAR voxels labelled *table*, before and after the model table is detected. Bottom-right: the resulting working-memory graph—robot, room, table and their epistemic affordances—with the table's FE and variance falling as the robot reaches the targeted viewpoints. Best seen in colour.

4 Experiments

As an ongoing project, we report preliminary results at two levels: the perception loop on the `room` localiser (validated against simulator ground truth) and the closed perception–action loop at the object level on the `table` agent. We begin with the `room` agent, running on the Shadow robot [17] as it autonomously explores a furnished room in Webots. The agent consumes the raw LiDAR stream and refines its pose belief $q = [x, y, \theta]^\top$ online while the controller drives the robot through a sequence of epistemic targets. Over a 1273 s, 352 m autonomous run the localiser updates at 19.1 Hz (median 0.8 ms per cycle), so perception comfortably keeps pace with the sensor.

Figure 4 makes the single perception–action loop of Sect. 3 explicit (a) and shows one cycle realised on this run (b). Because the motion prediction from

wheel odometry is usually accurate, an SDF quality gate lets the agent skip the gradient descent on 97% of frames (each 0.8 ms), reserving the ~ 81 ms Adam re-optimisation for the rare frame whose prediction no longer explains the incoming evidence. Each such moment is a *surprise event* (181 over the run, ~ 8.5 per minute): the variational free energy $\mathcal{F}$ spikes—up to 304 against a median of 0.07—and is re-minimised within ~ 1 s. Because the pose is by then well localised, the posterior variance σ_x^2 stays near its floor through the spike rather than collapsing: the surprise is absorbed without destabilising the belief. The agent thus does not drive $\mathcal{F}$ monotonically to zero; once well localised it sits near the floor, keeping the belief anchored across the 352 m trajectory by paying, and immediately repaying, a burst of surprise at each manoeuvre—the robot moves, becomes surprised, and re-represents.

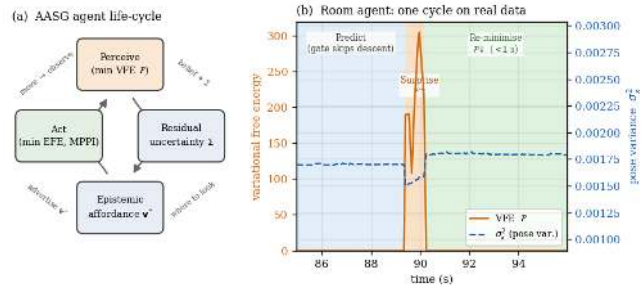


Fig. 4: (a) The Active-Inference life-cycle shared by all ASG concept agents: a single free-energy functional drives both perception (VFE) and action (EFE). (b) The `room` agent realising the loop on a real Webots run: a surprise spike in the free energy $\mathcal{F}$ on new evidence is re-minimised within ~ 1 s, while the posterior variance σ_x^2 stays near its floor—the surprise is absorbed without destabilising the already well-localised belief. Phases: *Predict* (an SDF quality gate skips the descent), *Surprise* ($\mathcal{F}$ rises on new evidence) and *Re-minimise* ($\mathcal{F}$ returns to the floor).

The resulting belief is also globally accurate. Figure 5 compares the estimated trajectory against Webots ground truth—read live through the simulator bridge—over four autonomous runs (6.6–10.1 m each, 5,468 poses). After a rigid SE(2) alignment of the room and world frames, the pooled per-pose position error has a median of 72 mm (p90 145 mm) and the heading error a median of 0.8°, tight across runs (67 ± 17 mm, 0.7 ± 0.2°). The metric is one of trajectory-shape consistency rather than an absolute offset, since estimate and ground truth live in different frames; two further logs in which the robot stayed static are excluded, as a stationary trajectory makes the SE(2) rotation ill-posed.

The same loop operates at the object level. Figure 6 follows the `table` agent through a 164 s heterogeneous run (a room, a table, three chairs and a bottle; its working memory is shown in Fig. 2) in which the centralised controller,

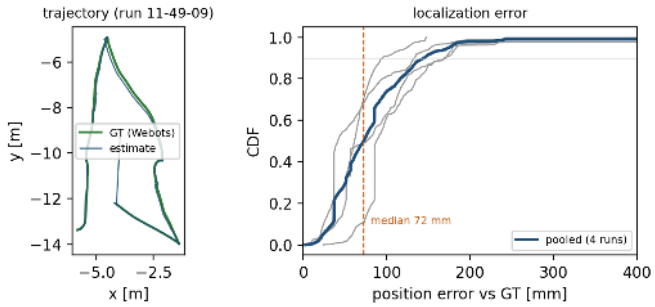


Fig. 5: Room-agent localisation vs. Webots ground truth, aggregated over four autonomous runs. (a) Estimated trajectory (blue) against ground truth (green) for a representative run. (b) Cumulative distribution of the per-pose position error after rigid SE(2) alignment: per-run (grey) and pooled (blue, median 72 mm). Heading error median 0.8°.

minimising EFE, selects the table as its active epistemic target at $t \approx 112\,\mathrm{s}$. While tracked passively the agent sits at a free-energy baseline of $\mathcal{F} \approx 2.47$. Once the table becomes the planning target the robot *moves to represent* it: the fresh viewpoints it gathers sharpen the 7-DoF posterior during the visit (σ_{c_x} contracts by 31%, the yaw σ_ψ by 25%) and drive the free energy down to $\mathcal{F} \approx 2.04$, 18% below the pre-target baseline. The object node thus reproduces the room-level signature of Fig. 4: acting on an epistemic affordance minimises the very free energy that scores the belief. A direct quantitative evaluation of the EFE-driven *selection* itself is left to future work; here the action loop is exercised through its measurable effect on the perceptual free energy it is designed to minimise.

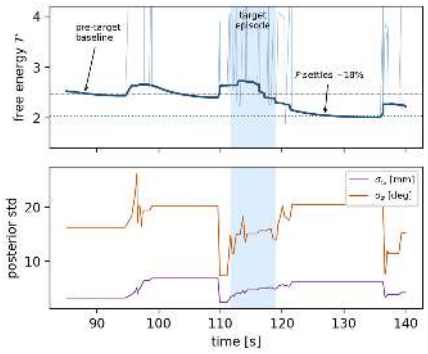


Fig. 6: Object-level free-energy minimisation on the `table` agent across one controller target episode (shaded) of a 164 s heterogeneous run. Top: the free energy $\mathcal{F}$ (light: per-cycle; bold: rolling median) settles 18% below the pre-target baseline (dashed) once the table becomes the planning target. Bottom: the posterior standard deviations (centre σ_{c_x}, yaw σ_ψ) contract as the robot gathers viewpoints. Same minimise-on-acting signature as the `room` localiser (Fig. 4), now on a 7-DoF object node.

5 Conclusion and Future Work

We have presented a reframing of working memory in robotics through Active Scene Graphs. By giving perception and action equal roles within the CORTEX architecture, we enable a self-constructing representation where the robot's motion is guided by the minimisation of uncertainty. The combination of variational inference over distance-field models and EFE-driven control via MPPI provides a robust and biologically inspired foundation for autonomous robot exploration and persistent world modelling. The next steps in this research will address three key aspects. First, the generalisation of the object models to concept classes that can be partially learnt on line. Second, the extension of the epistemic affordances idea to relational, inter-object, exploratory actions, and its combination with affordances derived from human-designed missions. A third improvement will introduce a structural mechanism to create hierarchies in the joint model. As a first step, we will remove the mean-field assumption taken at the object level and include inter-object constraints—e.g. doors are in walls—that will generate their own free energy and corresponding minimisation procedure; changes at this level will propagate down as precision re-weighting in the loss function.

Acknowledgments

This work was co-funded by the FEDER Project 0124_EUROAGE_MAS_4_E under the POCTEP Programme 2021–2027, and by the R&D&i project PID2022-137344OB-C31, supported by MICIU/AEI/10.13039/501100011033 and "FEDER,

Una manera de hacer Europa". Additional co-funding was provided by the European Union through the European Regional Development Fund (85) and by the Junta de Extremadura. The managing authority is the Ministerio de Hacienda (Spain). Grant GR24194 and by the SWEET Project, Horizon Europe Marie Skłodowska-Curie grant (Agreement No. 101168792).

References

1. Armeni, I., He, Z.Y., Gwak, J., Zamir, A.R., Fischer, M., Malik, J., Savarese, S.: 3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera (2019)
2. Bajcsy, R., Aloimonos, Y., Tsotsos, J.K.: Revisiting Active Perception (2016)
3. Bavle, H., Sanchez-Lopez, J.L., Shaheer, M., Civera, J., Voos, H.: Situational Graphs for Robot Navigation in Structured Indoor Environments (2022)
4. Buckley, C.L., Kim, C.S., McGregor, S., Seth, A.K.: The free energy principle for action and perception: A mathematical review. Journal of Mathematical Psychology **81**, 55–79 (2017)
5. Bustos, P., Manso, L., Bandera, A., Bandera, J., García-Varea, I., Martínez-Gómez, J.: The CORTEX cognitive robotics architecture: Use cases. Cognitive Systems Research **55** (2019)
6. Bustos García, P., Manso Argüelles, L.J., Bandera, A., Bandera, J.P., García-Varea, I., Martínez-Gómez, J.: CORTEX: A Novel Cognitive Architecture for Social Robots. In: Proceedings of the EUCognition Meeting on Cognitive Robot Architectures. Vienna, Austria (2016)
7. Fredriksson, S., Saradagi, A., Nikolakopoulos, G.: Robotic Exploration through Semantic Topometric Mapping. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). pp. 9404–9410. IEEE (2024)
8. Friston, K.: The free-energy principle: a unified brain theory? Nature Reviews Neuroscience (2010), bibTeX fields may need completion (volume/number/pages/doi).
9. Gu, Q., Kuwajerwala, A., Morin, S., Jatavallabhula, K.M., Sen, B., Agarwal, A., Rivera, C., Paul, W., Ellis, K., Chellappa, R., Gan, C., Melo, C.M.d., Tenenbaum, J.B., Torralba, A., Shkurti, F., Paull, L.: ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning (2023)
10. Hughes, N., Chang, Y., Hu, S., Talak, R., Abdulhai, R., Strader, J., Carlone, L.: Foundations of spatial perception for robotics: Hierarchical representations and real-time systems. The International Journal of Robotics Research p. 02783649241229725 (2024)
11. Kelley, T.: Standard model of mind: Episodic Memory. Procedia Computer Science (2018)
12. Lanillos, P., Meo, C., Pezzato, C., Meera, A.A., Baioumy, M., Ohata, W., Tschantz, A., Millidge, B., Wisse, M., Buckley, C.L., Tani, J.: Active Inference in Robotics and Artificial Agents: Survey and Challenges (Dec 2021). https://doi.org/10.48550/arXiv.2112.01871
13. Manso, L., Bachiller, P., Bustos, P., Núñez, P., Cintas, R., Calderita, L.: Robocomp: A tool-based robotics framework. In: Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., von Stryk, O. (eds.) Simulation, Modeling, and Programming for Autonomous Robots. pp. 251–262. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
14. Rosinol, A., Violette, A., Abate, M., Hughes, N., Chang, Y., Shi, J., Gupta, A., Carlone, L.: Kimera: from SLAM to Spatial Perception with 3D Dynamic Scene Graphs (2021)

15. Rotondi, D., Scaparro, F., Blum, H., Arras, K.O.: FunGraph: Functionality Aware 3D Scene Graphs for Language-Prompted Scene Interaction (Mar 2025). https://doi.org/10.48550/arXiv.2503.07909
16. Saucedo, M.A., Stathoulopoulos, N., Patel, A., Kanellakis, C., Nikolakopoulos, G.: Leveraging Computation of Expectation Models for Commonsense Affordance Estimation on 3D Scene Graphs. In: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 9797–9802 (2024)
17. Torrejón, A., Zapata, N., Bonilla, L., Bustos, P., Núñez, P.: Design and development of shadow: A cost-effective mobile social robot for human-following applications. Electronics **13**(17) (2024)
18. Trafton, G., Hiatt, L., Harrison, A., Tanborello, F., Khemlani, S., Schultz, A.: ACT-R/E: An Embodied Cognitive Architecture for Human-Robot Interaction. Journal of Human-Robot Interaction **2**(1), 30–55 (2013)
19. Williams, G., Aldrich, A., Theodorou, E.A.: Model predictive path integral control: From theory to parallel computation (2017), check this entry matches the MPPI paper you intend to cite (venue/pages/doi).
20. ZeroC, Inc.: Internet communications engine (ice). https://www.zeroc.com/ice (2026), accessed: 2026-06-11

Multi-Sensor Fusion and Temporal Synchronization for Human Activity Recognition in Human–Robot Collaboration in Precision Agriculture

Nelson Broyer Paco Chipana[1][0000-0002-9147-3334], Pedro Miguel Núñez Trujillo[1][0000-0002-3615-8833], Pablo Bustos García De Castro[1][0000-0003-3202-2493], and Luis Vicente Calderita Estévez[1][0000-0003-0784-4102]

[1] Robolab, Robotics and Artificial Vision, University of Extremadura, Cáceres, Spain
nelson.bpch@unex.es
[2] pnuntru@unex.es
[3] pbustos@unex.es
[4] lvcalderita@unex.es

Abstract. Precision agriculture increasingly relies on collaborative systems in which human workers and autonomous robots share operational spaces, each performing tasks that complement the other's capabilities. Effective collaboration requires robots to perceive and interpret the actions of nearby workers in real time. This doctoral research is centered on multi-sensor fusion for robotic perception in human–robot collaboration (HRC), with temporal synchronization of heterogeneous, asynchronous sensor streams—cameras, LiDAR, and inertial measurement units (IMUs). Human activity recognition (HAR), built primarily from methods available in the literature, is the main application through which the fusion and synchronization framework is instantiated and evaluated. The HAR module combines human pose estimators and lightweight machine learning classifiers to infer agricultural workers' activities at inference time. Vision-Language Models (VLMs) will be evaluated as potential components to enhance activity recognition in agricultural scenarios, with their integration to be determined based on accuracy performance under agricultural conditions and compatibility with temporal synchronization requirements. The synchronization mechanism specifically addresses the alignment of high-rate sensor data with the variable-latency, non-deterministic outputs of inference modules, including potential VLM-based semantic processing. The framework will be implemented using the Robocomp framework, validated in an intermediate stage through simulation, and finally validated through field experiments on a real robotic platform, targeting degradation conditions character-

istic of agricultural settings such as illumination variability and terrain irregularity.

Keywords: Multi-sensor fusion · Temporal synchronization · Human activity recognition · Human–robot collaboration · Precision agriculture · Vision-Language Models · Autonomous navigation

1 Introduction and Problem Statement

Precision agriculture is increasingly incorporating robotic systems capable of autonomous operation in outdoor field conditions [9,16]. Robots offer advantages such as continuous operation and consistent performance, but do not yet match human workers in every task; selective harvesting of ripe fruit, for instance, still benefits from human dexterity and perceptual judgment [17,14]. A practical paradigm is therefore one of structured collaboration, in which humans and autonomous ground vehicles (UGVs) act as complementary agents within a shared workspace [5].

For this collaboration to be safe and productive, a UGV operating near an agricultural worker should identify not only the worker's location but also the activity being performed—for example, carrying a load, inspecting a crop row, or resting—so that socially-aware navigation decisions informed by worker activity recognition can be adapted dynamically [12]. This requires a perception pipeline that fuses multiple sensor modalities reliably under the dynamic conditions of real agricultural environments, and that supports a HAR module operating on top of it.

Agricultural environments in both open-field and greenhouse scenarios present perception challenges distinct from controlled indoor collaborative robotics: (i) dynamic occlusions caused by dense vegetation—workers disappear behind foliage only to reappear moments later, creating temporal gaps in visual perception while LiDAR and IMU continue sensing, requiring temporal alignment across modalities with incomplete information; (ii) extreme illumination variability—direct sunlight, deep shadows, specular reflections, and rapid cloud-induced changes occur simultaneously and locally within the visual field, degrading real-time camera-based activity inference; (iii) irregular and unpredictable human movement patterns—workers alternate rapidly between walking, stopping, crouching, bending, and load-carrying within seconds, violating constant-velocity assumptions common in motion models; and (iv) safety-critical operation requiring robust activity inference to maintain safe collaboration distances and react appropriately to worker movements. Additionally, GPS signals are degraded under dense vegetation, uneven terrain affects wheel odometry, and these conditions compound sensor fusion challenges [19,10]. These limitations motivate GPS-independent multi-sensor fusion strategies that combine cameras, LiDAR, and IMUs, since combining complementary modalities can produce more continuous and robust estimates than a single sensor [21]. This doctoral research

addresses precisely this GPS-denied scenario, with the experimental scope restricted to camera, LiDAR, and IMU modalities. Fusing these heterogeneous sensor streams, however, requires temporal synchronization: sensors operate at different sampling rates and latencies, and without proper alignment, fused estimates may combine data from different moments in time, degrading localization, mapping, and activity recognition [6]. This challenge is particularly acute when visual occlusions cause temporary loss of human tracking, while asynchronous inference modules (such as VLMs) introduce variable-latency that must be temporally aligned with sensor observations for consistent activity inference. Synchronizing sensor data with non-deterministic, high-latency inference is the central technical challenge addressed by this research.

This issue is particularly relevant when high-latency, non-deterministic inference modules, such as VLMs applied to camera imagery, are incorporated into the perception pipeline, since their inference time can exceed the sampling period of higher-rate sensors such as LiDAR or IMUs. Rather than treating HAR, sensor fusion, and temporal synchronization as three separate problems, this doctoral research is centered on multi-sensor fusion for perception, with temporal synchronization as its core methodological contribution; HAR constitutes the primary application through which the fusion and synchronization framework is instantiated and evaluated, relying where possible on methods already available in the literature rather than building every component from scratch.

2 State of the Art

2.1 Human–Robot Collaboration in Agricultural Environments

Comai et al. [5] propose an ontology-based cooperative framework in which agrobots adjust their autonomy level according to task and environmental conditions. Shereuzhev et al. [17] model collaborative multi-robot systems for fruit harvesting using game-theoretic formulations, reporting efficiency gains from dynamic task redistribution among human operators and robots, validated in simulation. Pal et al. [14] present a vision-based pipeline that detects, tracks, and classifies fruit-picker activities using YOLOv5, Deep-SORT, and LSTM networks, enabling a mobile robot to navigate toward workers and provide logistic assistance.

2.2 Human Activity Recognition for Agricultural HRC

HAR research in this domain has used both wearable sensors and vision-based methods. Anagnostis et al. [1] applied LSTM networks to data from body-worn IMUs during load-carrying tasks, reporting 85.6 % accuracy in a farm deployment. Mekruksavanich and Jitpattanakul [11] proposed a lightweight causal convolutional architecture combining accelerometer, gyroscope, and magnetometer signals for real-time edge deployment. Benos et al. [4] integrated SHAP-based explainability with an XGBoost classifier, identifying torso-mounted IMUs as an

informative sensor placement. On the vision side, Moysiadis et al. [12] used OpenPose skeleton extraction and LSTM classification to map worker movements onto UGV commands in an orchard, and Pal et al. [14] combine activity classification with goal-location estimation for multi-worker scenarios. These methods provide a basis of existing, reusable components for the HAR module proposed in this work.

2.3 Multi-Sensor Fusion for Agricultural Navigation

Yang et al. [19] survey multi-sensor fusion-based agricultural navigation, noting that Kalman filter variants remain the most widely used fusion algorithms. Zhao et al. [21] present an adaptive LiDAR–visual–inertial odometry framework for GPS-denied agricultural scenarios. Liang et al. [10] address GNSS noise caused by canopy multi-path effects through Gaussian Mixture Model-based fusion with stereo visual odometry. For navigation-line extraction, Ban et al. [2] fuse camera, LiDAR, and IMU data for maize-row detection, and Shi et al. [18] fuse vision and 2D LiDAR for orchard-row navigation. For obstacle detection, Zhang et al. [20] propose a LiDAR-camera fusion pipeline with cascaded ground filtering and Kalman-based tracking. The GNSS-related works above ([19,10]) illustrate strategies to mitigate GPS degradation; the present research instead targets the GPS-independent case, restricting its experimental modalities to cameras, LiDAR, and IMUs, in line with [21,2,18,20].

2.4 Temporal Synchronization and Data Fusion in Agriculture

Temporal synchronization has received comparatively less attention in agricultural robotics. Dai et al. [6] propose a hardware-free spatiotemporal synchronization method for UAV LiDAR-camera systems based on terrain contours. Samadzadegan et al. [15] review multi-platform, multi-sensor data fusion, identifying temporal alignment as one of the central challenges alongside spatial and radiometric consistency. Barbedo [3] reviews data fusion in agriculture broadly; Kumar [13] surveys multi-modal sensing covering temporal and spatial alignment and edge computing; Lagiewska and Panek-Chwastyk [9] review the integration of remote sensing with autonomous ground robots; Khan et al. [8] use Dempster-Shafer decision-level fusion for sensor-fault handling in weed detection; and Jeyabalan [7] compares LiDAR-fusion EKF, V-SLAM, and a hybrid approach for GPS-denied UAV navigation in orchard canopies. None of these works jointly addresses synchronization for non-deterministic, high-latency inference modules such as VLMs together with high-rate sensor fusion, which is the gap targeted by this research.

3 Objectives

3.1 General Objective

To design, implement, and evaluate a multi-sensor fusion framework with explicit temporal synchronization for context-aware robotic perception in human–robot

collaboration in agricultural environments, applying it to support human activity recognition and socially-aware autonomous navigation using camera, LiDAR, and IMU data.

3.2 Specific Objectives

1. To characterize sources of uncertainty affecting camera, LiDAR, and IMU-based robotic perception, localization, and activity inference in agricultural environments, including dynamic occlusions and illumination variability.
2. To develop a temporal synchronization mechanism that aligns heterogeneous sensor streams and accounts for variable inference delays introduced by high-latency modules such as VLMs, as the core methodological contribution of this research.
3. To design a HAR module, built primarily from methods available in the literature and combining multi-view camera-based pose estimation and lightweight machine learning classifiers, to infer worker activities. Vision-Language Models (VLMs) will be investigated as optional components, with their specific role and feasibility to be determined through inference latency and performance evaluation.
4. To integrate the HAR module and the synchronization mechanism within a multi-sensor, GPS-independent navigation architecture operating with socially-aware behavior, so that navigation responses are adapted based on inferred worker activities and safe collaboration distances are maintained.
5. To evaluate the proposed framework through three complementary stages: HAR validation against benchmarks, intermediate simulation-based evaluation under controlled degradation, and field experiments with a real robot—measuring HAR accuracy, localization error, temporal consistency, and navigation behaviour, and quantifying synchronization impact relative to non-synchronized baselines.

4 Research Hypothesis

Explicitly modeling and compensating temporal misalignment among heterogeneous, asynchronous sensor streams (cameras, LiDAR, IMUs) and variable-latency inference modules (e.g., VLM-based semantic processing) within a multi-sensor fusion perception pipeline will produce measurable improvements, relative to a non-synchronized baseline, in (i) HAR accuracy, (ii) localization accuracy, and (iii) navigation performance, under the illumination variability and terrain irregularity characteristic of agricultural environments. Moreover, the synchronization mechanism must remain effective under conditions of partial occlusion, where visual tracking is interrupted but other modalities continue to provide information.

5 Methodology Summary

The research follows five sequential phases:

1. **Uncertainty Characterization and Initial HAR Validation:** profiling of temporal characteristics (sampling rate, latency, jitter, dropout) of camera, LiDAR, and IMU data using existing agricultural datasets, together with independent validation of candidate HAR model components (pose estimators, classifiers, etc.) against literature benchmarks. This phase also defines the maximum tolerable latency budget for inference modules and the performance criteria that Vision-Language Models (VLMs) must satisfy to be feasible for integration. The original HAR–navigation dataset is collected later, in Phase 5.
2. **Multi-View Camera-Based HAR Module:** a configuration combining multi-camera pose estimation and lightweight ML classifiers (LSTM, 1D-CNN, or gradient-boosted trees), with SHAP-based explainability. Vision-Language Models (VLMs) will be investigated for their capability to recognize human activities in agricultural scenarios, evaluated against the latency budget and performance criteria established in Phase 1; integration will proceed only if they demonstrate sufficient accuracy under agricultural conditions (occlusions, illumination variability) *and* their inference latency is compatible with synchronization requirements. The HAR architecture is adapted to agricultural scenarios, particularly addressing robustness to dynamic occlusions and illumination changes. To be refined according to Phase 1 results, prioritizing lightweight classifiers and sensor-based features if VLM performance does not meet agricultural accuracy thresholds.
3. **Temporal Synchronization Mechanism:** the core contribution of the thesis. Addresses the temporal alignment of heterogeneous, asynchronous sensor streams (camera, LiDAR, IMU) operating at different sampling rates and with variable latencies. The framework combines: (i) timestamp-based alignment across Robocomp components for low-level sensor synchronization, and (ii) a multi-sensor buffer that stores observations with temporal metadata, enabling post-hoc alignment of inference module outputs (such as pose estimation or activity recognition). Temporal prediction techniques (e.g., Kalman filtering, particle filtering) will be investigated to compensate for variable delays introduced by computationally expensive inference modules, if determined necessary through Phase 1 and 2 results. A model will be developed relating maximum tolerable desynchronization to robot speed, sensor rates, and safety distance requirements. Evaluated through direct metrics (timestamp deviation, jitter) and indirect metrics (impact on HAR accuracy, localization error, and navigation performance).
4. **Integration and System Architecture:** integration of HAR and synchronization within a Robocomp-based socially-aware navigation stack using LiDAR–visual–inertial odometry (GPS-independent) and LiDAR–camera fusion for obstacle detection. The simulator platform will be selected based on compatibility with Robocomp and multi-sensor simulation requirements, without commitment to a specific simulator at this stage. The modular architecture allows independent testing and replacement of components.
5. **Evaluation:** three stages—(a) independent validation of integrated HAR system against literature benchmarks; (b) simulation-based validation of the

integrated system under controlled degradation conditions; and (c) field experiments with a real robot (current platform evaluation: Clearpath Husky or equivalent ground vehicle platform, pending final configuration and procurement decisions), which also yields the synchronized multi-sensor dataset for internal validation and future reproducibility research. Metrics: localization RMSE, cross-modal timestamp deviation/jitter, HAR accuracy, navigation success rate, collision-free operation rate, and the indirect synchronization-impact metrics. The dataset will not be presented as a primary contribution but will be made available to support reproducibility and future research.

6 Expected Results and Contributions

- **Temporal synchronization mechanism:** the central contribution of this work—an approach to temporal alignment for multi-sensor pipelines combining heterogeneous sensors with variable-latency inference outputs, implemented and evaluated in an agricultural robotic context, including its measured impact on HAR, localization, and navigation performance.
- **Uncertainty characterization:** an analysis of sensor failure modes and their impact on perception and navigation in agricultural environments, with emphasis on occlusion dynamics and illumination variability, restricted to camera, LiDAR, and IMU modalities.
- **Agricultural HAR system:** a HAR architecture built primarily from existing literature methods and adapted for agricultural scenarios, addressing robustness to occlusions and illumination changes, evaluated for accuracy, latency, and explainability on benchmark datasets.
- **Socially-aware, GPS-independent navigation architecture:** a navigation stack based on camera, LiDAR, and IMU fusion that incorporates HAR outputs as inputs to socially-aware navigation decisions, enabling activity-aware collaboration.
- **Multi-sensor synchronized dataset:** an original multi-modal agricultural dataset with synchronized sensor, HAR, and navigation data, collected once the integrated system is available, intended to support reproducibility and future research.
- **Publications:** journal articles and conference papers reporting results in agricultural robotics, sensor fusion, and temporal synchronization.

7 Conclusion

This proposal presents a structured plan centered on multi-sensor fusion and temporal synchronization for perception in human–robot collaboration in precision agriculture. Temporal synchronization of heterogeneous, asynchronous sensor streams with variable-latency inference outputs constitutes the core methodological contribution. Human activity recognition, built primarily from existing literature methods, serves as the main application through which the framework

is instantiated and evaluated, and is itself validated independently against literature benchmark datasets before integration. The resulting framework, restricted in experimental scope to camera, LiDAR, and IMU modalities, will be implemented using Robocomp, validated in an intermediate stage through simulation, and finally validated through field experiments on a real robotic platform. The research builds on existing literature in agricultural HAR, sensor fusion, and temporal synchronization, addressing aspects that, based on the literature reviewed, remain only partially addressed. The expected outcomes are intended to contribute to the design of agricultural robots capable of operating alongside human workers under the conditions of real field environments.

References

1. Anagnostis, A., Benos, L., Tsaopoulos, D., Tagarakis, A., Tsolakis, N., Bochtis, D.: Human activity recognition through recurrent neural networks for human–robot interaction in agriculture. Applied Sciences **11**, 2188 (2021). https://doi.org/10.3390/app11052188
2. Ban, C., Wang, L., Chi, R., Su, T., Ma, Y.: A camera-LiDAR-IMU fusion method for real-time extraction of navigation line between maize field rows. Computers and Electronics in Agriculture **223**, 109114 (2024). https://doi.org/10.1016/j.compag.2024.109114
3. Barbedo, J.G.A.: Data fusion in agriculture: Resolving ambiguities and closing data gaps. Sensors **22**, 2285 (2022). https://doi.org/10.3390/s22062285
4. Benos, L., Tsaopoulos, D., Tagarakis, A.C., Kateris, D., Busato, P., Bochtis, D.: Explainable AI-enhanced human activity recognition for human–robot collaboration in agriculture. Applied Sciences **15**, 650 (2025). https://doi.org/10.3390/app15020650
5. Comai, S., Fugini, M., Gusmeroli, S., Salice, F.: An approach to cooperative human-robot teaming in smart agriculture. In: Proceedings of the 32nd IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (2024). https://doi.org/10.1109/WETICE64632.2024.00012
6. Dai, K., Zhu, J., Hou, H., Li, Q., Ma, X., Yu, H.: A LiDAR–Camera spatiotemporal synchronization method for unmanned aerial vehicle-based ground target perception. IEEE Transactions on Instrumentation and Measurement **74**, 1008715 (2025). https://doi.org/10.1109/TIM.2025.3558243
7. Jeyabalan, K.: Autonomous navigation algorithms for agricultural drones in dense canopies: A comparative study of LiDAR-fusion and vision-based approaches. World Journal of Engineering and Technology **1**(2), 1–10 (2025)
8. Khan, M.N., Rahi, A., Hasan, M.A., Anwar, S.: Decision-level multi-sensor fusion to improve limitations of single-camera-based CNN classification in precision farming: Application in weed detection. Computation **13**, 174 (2025). https://doi.org/10.3390/computation13070174
9. Lagiewska, M., Panek-Chwastyk, E.: Integrating remote sensing and autonomous robotics in precision agriculture: Current applications and workflow challenges. Agronomy **15**, 2314 (2025). https://doi.org/10.3390/agronomy15102314
10. Liang, S., Zhao, W., Lin, N., Huang, Y.: Adaptive fusion positioning based on Gaussian mixture model for GNSS-RTK and stereo camera in arboretum environments. Agronomy **13**, 1982 (2023). https://doi.org/10.3390/agronomy13081982

11. Mekruksavanich, S., Jitpattanakul, A.: Efficient wearable sensor-based activity recognition for human–robot collaboration in agricultural environments. Informatics **12**, 115 (2025). https://doi.org/10.3390/informatics12040115
12. Moysiadis, V., Benos, L., Karras, G., Kateris, D., Peruzzi, A., Berruto, R., Papageorgiou, E., Bochtis, D.: Human–robot interaction through dynamic movement recognition for agricultural environments. AgriEngineering **6**, 2494–2512 (2024). https://doi.org/10.3390/agriengineering6030146
13. Nagireddy, S.K.: Sensor-driven autonomy in agriculture: A multi-modal approach to precision farming. Journal of Computer Science and Technology Studies **7**(7), 979–986 (2025). https://doi.org/10.32996/jcsts.2025.7.7.108
14. Pal, A., Leite, A.C., From, P.J.: A novel end-to-end vision-based architecture for agricultural human–robot collaboration in fruit picking operations. Robotics and Autonomous Systems **172**, 104567 (2024). https://doi.org/10.1016/j.robot.2023.104567
15. Samadzadegan, F., Toosi, A., Javan, F.D.: A critical review on multi-sensor and multi-platform remote sensing data fusion approaches: Current status and prospects. International Journal of Remote Sensing **46**(3), 1327–1402 (2025). https://doi.org/10.1080/01431161.2024.2429784
16. Shalash, O., Emad, A., Fathy, F., Alzogby, A., Sallam, M., Naser, E., El-Sayed, M., Khatab, E.: Fusion of robotics, AI, and thermal imaging technologies for intelligent precision agriculture systems. Sensors **25**, 6844 (2025). https://doi.org/10.3390/s25226844
17. Shereuzhev, M.A., Dyshekov, A.I., Devyatkin, F.V.: Adoption of collaborative robotics in fruit harvesting. Agricultural Machinery and Technologies **19**(4), 66–74 (2025). https://doi.org/10.22314/2073-7599-2025-19-4-66-74
18. Shi, Z., Bai, Z., Yi, K., Qiu, B., Dong, X., Wang, Q., Jiang, C., Zhang, X., Huang, X.: Vision and 2D LiDAR fusion-based navigation line extraction for autonomous agricultural robots in dense pomegranate orchards. Sensors **25**, 5432 (2025). https://doi.org/10.3390/s25175432
19. Yang, J., Zulkarnain, N., Ibrahim, M.F., Nordin, I.N.A.M., Vaghefi, S.A.: A comprehensive review of agricultural ground automatic navigation systems based on multi-sensor fusion. IEEE Access **13**, 168159–168198 (2025). https://doi.org/10.1109/ACCESS.2025.3612811
20. Zhang, W., Lin, Y., Huang, Q., Ding, F., Luo, X., Zhang, Z., Hu, L., Deng, Y., Wu, J., Li, J.: Robust long-range obstacle detection in farmland via LiDAR-Camera fusion and cascaded ground filtering. Computers and Electronics in Agriculture **250**, 111886 (2026). https://doi.org/10.1016/j.compag.2026.111886
21. Zhao, Z., Zhang, Y., Long, L., Lu, Z., Shi, J.: Efficient and adaptive Lidar–Visual–Inertial odometry for agricultural unmanned ground vehicle. International Journal of Advanced Robotic Systems pp. 1–15 (2022). https://doi.org/10.1177/17298806221094925

Explainability as debugging tool for AI-powered robotics

Jorge Rodriguez[1], Luis Bote-Curiel[1][0000−0001−8845−2834], David Roldán[1][0000−0001−7049−7460], and José M. Cañas[1][0000−0003−4179−2211]

Escuela Ingeniería de Fuenlabrada, Universidad Rey Juan Carlos, Spain

Abstract. Robotics applications increasingly include Machine Learning components, elements which have been programmed with DeepLearning or Deep Reinforcement Learning techniques. They typically provide high robustness in real world conditions. But they are difficult to debug as the behavior of the neural networks is somehow hidden in the numerical values of a huge number of weights. Introducing instrospection and explainability in these AI components may help to understand the behavior of the neural network and to debug it if needed. This paper presents a preliminary use of two tools for interpretable AI, SHAP and Grad-Cam, to debug a robotics application in autonomous driving domain, the visual follow-lane task. An end-to-end neural model has been trained for it. Intrepretable AI tools help to discover which parts of the input images the model is paying attention to and so affect its control output. This introspection has suggested the cropping of the input images and the inclusion in the training dataset of more variety of shadows, which have improved the performance of the end2end model when driving the autonomous car.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

The increasing use of Artificial Intelligence (AI) techniques in robotics, particularly Machine Learning and especially its subset Deep Learning, has enabled significant advances in perception, decision-making, and control. However, many of these methods behave as black boxes, providing high performance but little insight into the reasons behind their predictions. This lack of transparency is particularly problematic in robotic systems, where model outputs can ultimately be translated into physical actions in the real world.

This opacity has motivated a growing research area concerned with making Machine Learning models more understandable, commonly discussed under the labels of Explainable AI (XAI) and Interpretable Machine Learning (IML), which are associated with the terms of explainability and interpretability, respectively. However, these two terms do not have universally accepted definitions and are often used in different ways across the literature [6, 13, 9]. Some authors draw a clear distinction between them. For example, Rudin [21] argues that interpretability refers to models that are inherently transparent while explainability

concerns post-hoc methods applied to opaque models. Lipton [13], however, differentiates them by the questions they address, where interpretability asks how the model works and explainability asks what else the model can tell us. Other authors adopt a more pragmatic view treating both terms as essentially interchangeable and viewing them as any technique that extracts relevant knowledge from a Machine Learning model [17, 16]. In this paper, we adopt this practical perspective, and although we primarily use the term explainability, we treat both terms interchangeably to refer to methods that help developers inspect, understand, and debug the behavior of Machine Learning models.

This concept of explainability is especially relevant in visual robot navigation and autonomous driving [24, 1]. Among the different approaches to the latter, end-to-end visual control is particularly opaque, as a Deep Neural Network directly maps camera images to control commands such as steering actions. While this approach reduces the need for hand-crafted pipelines, it also makes it difficult to determine whether the model is relying on meaningful visual cues, such as lane markings and road boundaries, or on spurious elements present in the training data. Explainability methods are commonly applied to these already trained models to understand which parts of the input drive their predictions.

In this work, we explore the use of explainability not only to understand the decisions of already trained models, but as a debugging tool for an AI-powered robotic system. We apply explainability during development to analyze what the model has learned, identify possible failure causes, and guide design decisions. In particular, we focus on a vision-based task for autonomous driving and analyze how explainability methods can reveal how image regions, cropping strategies, and visual artifacts such as shadows or spurious objects influence the behavior of the model, guiding decisions such as input image cropping or training dataset diverse enrichment to improve model performance.

2 Related works

The growing adoption of Machine Learning and Deep Learning in safety-critical applications has motivated substantial research on explainability. In the specific context of autonomous driving, Zablocki et al. [24] review explainability methods for end-to-end vision-based driving systems, framing the discussion around four key questions: who needs explanations, why they are needed, what kind of explanations can be generated, and when they should be delivered. Atakishiyev et al. [1] offer a broader overview of XAI approaches for autonomous vehicles, proposing a conceptual framework that integrates explainability with societal and regulatory requirements. More recently, Kuznietsov et al. [12] present a systematic review focusing on how XAI techniques contribute to safety and trustworthiness in autonomous driving, proposing the SafeX framework for modular integration of explainability into driving systems. These surveys consistently highlight the tension between the high performance of Deep Learning models and the limited understanding of their internal decision processes.

A foundational line of work in this area is the PilotNet [3] of NVIDIA, one of the earliest demonstrations of end-to-end Deep Learning model for autonomous steering. PilotNet maps raw camera images directly to steering angles using a Convolutional Neural Network (CNN) trained via Imitation Learning from human driving data. To understand what PilotNet learns, Bojarski et al. [4] developed a visualization method that identifies which input pixels most influence the steering prediction, showing that the model learns to attend to lane markings, road edges, and even subtle cues such as roadside vegetation. This work was further refined with VisualBackProp [2], a computationally efficient saliency method that propagates and combines averaged feature maps from successive convolutional layers back to the input space. VisualBackProp was designed as a verification tool for CNN-based steering systems, allowing developers to visually confirm that the network attends to relevant traffic elements rather than spurious image regions, while being fast enough to run in real time alongside inference.

Beyond saliency-based post-hoc methods, other works have explored attention mechanisms as a means of building explainability into the model architecture itself. Kim and Canny [10] propose a causal visual attention model for end-to-end driving that learns to focus on task-relevant image regions during steering prediction. Their approach goes beyond standard saliency maps by filtering attention through a causal lens, distinguishing regions that merely correlate with the output from those that causally influence it. Similarly, Kim et al. [11] extend this idea by generating textual explanations alongside driving decisions, combining visual attention with natural language to produce richer and more human-interpretable justifications for vehicle behavior. Cultrera et al. [5] present a conditional Imitation Learning architecture with an end-to-end visual attention model, showing through ablation studies that the integration of attention improves both driving performance and explainability in the Car Learning to Act (CARLA) simulator [7].

A closely related but less explored direction is the use of explainability not merely to understand a trained model, but to actively guide its improvement during development. This perspective treats saliency maps and feature attributions as diagnostic signals that can reveal dataset biases, input preprocessing issues, or unintended learned correlations. Mikołajczyk-Bareła [15] addresses this direction in the context of medical imaging, proposing a global explanation method for semi-automatic bias discovery and combining it with targeted data augmentation strategies to mitigate spurious correlations. Their Attribution Feedback approach fine-tunes models using an attribution loss that teaches the Neural Network to ignore irrelevant input regions, demonstrating that explainability can be integrated into the training loop itself. In the automotive industry, El Houd et al. [8] propose a hybrid method that combines model prediction scores with Gradient-weighted Class Activation Mapping (Grad-CAM) heatmaps to improve the accuracy of welding seam defect classification, showing that explainability can directly enhance model performance rather than serving only as a post-hoc analysis tool. Nowak et al. [18] demonstrate how interpreting

attention heatmaps in an object detection system for driver assistance can guide dataset augmentation, leading to improved detection results.

These works collectively illustrate a shift from treating explainability as a passive interpretive layer toward using it as an active component of the model development cycle. However, the application of this debugging-oriented use of explainability to end-to-end visual control in robotics remains underexplored. Most existing studies apply saliency methods after training as a verification step, but do not systematically leverage the resulting insights to inform design decisions. Our work contributes to closing this gap by using explainability methods, specifically SHapley Additive exPlanations (SHAP) and Grad-CAM, as iterative debugging tools during the development of a vision-based autonomous driving system for the follow-lane task.

3 Explainability tools

A central goal of explainability is to understand why a model produces the outputs it does. This can be approached at different levels of granularity, namely global and local. Global methods characterize the overall behavior of the model across the dataset, for example by identifying which features are generally most influential, while local methods explain individual predictions by identifying which parts of a specific input were most relevant to a particular output [16].

Another important distinction concerns whether explanations are obtained from models that are interpretable by design or from post-hoc analysis. Intrinsic approaches rely on models whose structure is directly understandable, whereas post-hoc methods are applied after training to analyze the behavior of an already learned model [16]. Post-hoc methods may further differ in their model dependence. Model-agnostic methods treat the model as a black box and can be applied to different architectures, whereas model-specific methods exploit internal model information.

A widely used family of post-hoc local methods is feature attribution, which assigns a relevance score to each input feature indicating how much it contributed, positively or negatively, to the prediction [16]. The nature of these features depends on the input modality: for tabular data they correspond to individual variables, for text to words or tokens, and for images to pixels or groups of pixels. When applied to image inputs, feature attribution is commonly referred to as pixel attribution, and the resulting explanations take the form of saliency maps or heatmaps, that is, visual overlays that highlight the image regions most influential to the prediction.

Within pixel attribution, two main families can be distinguished based on how relevance scores are computed [16]. Perturbation-based methods, such as SHAP and Local Interpretable Model-agnostic Explanations (LIME), systematically modify, mask or occlude parts of the input and observe the effect on the output. They are generally model-agnostic and can be applied to any architecture. Gradient-based methods, such as Vanilla Gradient, SmoothGrad, and Grad-CAM, instead exploit the gradients computed during backpropagation to

estimate the sensitivity of the output to the input or to intermediate feature maps. They require access to the internal computations of differentiable models and are usually more efficient to compute.

3.1 SHAP

SHAP [14] is a perturbation-based feature attribution method grounded in cooperative game theory. It builds on the concept of Shapley values [23], originally proposed as a way to fairly distribute the total payoff of a cooperative game among its players according to their individual contributions. In the context of Machine Learning, SHAP adapts this framework by treating each input feature as a player and the model prediction as the payoff to be distributed. Each feature receives a SHAP value that represents its contribution to the difference between the actual prediction and the average prediction across the dataset.

More formally, for a given input, SHAP constructs a local additive explanation model of the form $g(z) = \phi_0 + \sum_{j=1}^{M} \phi_j z_j$, where $z_j \in \{0, 1\}$ indicates whether feature j is present or absent, ϕ_0 is the base value corresponding to the average model output, and each ϕ_j is the Shapley value for feature j. These values satisfy desirable theoretical properties such as local accuracy, consistency, and fair distribution of the prediction among features [14].

Since computing exact Shapley values requires evaluating all possible subsets of features, which grows exponentially with the number of features, several estimation strategies have been proposed. KernelSHAP [14] preserves the model-agnostic nature of the framework by approximating Shapley values through a weighted local linear regression. Other variants, such as DeepSHAP and GradientSHAP, trade this model-agnostic property for computational efficiency by exploiting the internal structure of Neural Networks [14].

When applied to image data, SHAP values can be computed at different levels of resolution depending on the estimation strategy. Some approaches operate directly at the pixel level, while others group pixels into superpixels that serve as the units of analysis, reducing the computational cost of the estimation. In both cases, the result is a map of SHAP values over the image that indicates which regions contributed positively or negatively to the prediction.

3.2 Grad-CAM

Grad-CAM [22] is a gradient-based pixel attribution method designed to produce visual explanations for CNN models. It is a generalization of Class Activation Mapping (CAM) [25], extending its applicability to any CNN-based architecture, including those with fully connected layers, without requiring re-training or architectural modifications.

The key idea behind Grad-CAM is to leverage the spatial information retained in convolutional feature maps, which is progressively lost in fully connected layers. Given a target output y^c (e.g., a class score in classification or a specific output in regression), Grad-CAM computes the gradients of y^c with respect to the feature map activations A^k of a selected convolutional layer. These

gradients are globally averaged over all spatial positions to obtain an importance weight α_k^c for each feature map k:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{1}$$

where Z is the number of spatial positions in the feature map. The Grad-CAM localization map is then computed as a weighted combination of the feature maps, followed by a ReLU activation to retain only positive contributions:

$$L_{\text{Grad-CAM}}^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right) \tag{2}$$

The resulting map is a coarse heatmap of the same spatial dimensions as the selected convolutional layer. The choice of convolutional layer involves a trade-off between semantic level and spatial resolution, and while the last convolutional layer is a common choice, other layers can also be used. Since Grad-CAM operates on internal feature maps rather than on input pixels directly, different target outputs yield different localization maps for the same input image, allowing developers to inspect which regions are relevant for each specific prediction.

4 Deep Learning based End-to-end visual control for autonomous driving

In this paper, within the autonomous driving application domain, we focus on the visual follow-lane task. The autonomous car is equipped with a front-facing onboard camera and two actuators: the steering wheel and the accelerator pedal. For the experiments we use the CARLA simulator [7], which has become the de facto standard for research in urban autonomous driving (Fig. 1).

Fig. 1. CARLA simulation environment with the vehicle used in the experiments.

The follow-lane task can be addressed by explicitly programming an image processing algorithm to detect the lane combined with a PID controller for reactive decision making. However, it can also be solved using a Machine Learning approach based on Behavior Cloning, a form of Imitation Learning, with an end-to-end neural model [20, 19]. Following this approach, the solution is built in three steps: (1) collecting a supervised dataset while an expert properly drives the car, (2) selecting a neural network architecture, and (3) training the model with the collected dataset. At inference time, at each control iteration, the network receives the current camera image as input and produces the steering and throttle commands that are sent to the actuators.

4.1 Dataset

The training dataset was recorded on CARLA Town01 by an expert driver using a physical steering wheel setup (Fig. 2). The dataset contains approximately 28 000 samples and was designed to be large, varied, and balanced. Its composition is approximately 32–38% straight driving, 26–28% right turns, 26–28% left turns, and 10–12% recovery maneuvers. Recovery maneuvers, in which the driver corrects the trajectory from non-centered positions back to the lane center, are essential for the model to learn how to recover from deviations during inference rather than only learning to drive in ideal conditions.

Fig. 2. Physical steering wheel setup used for expert dataset recording.

4.2 Network architecture

As network architecture we selected PilotNet [3], the CNN model introduced by NVIDIA for end-to-end autonomous steering described in Section 2. Although more recent and complex architectures exist, PilotNet provides a well-established and sufficiently capable baseline for the follow-lane task. Since the focus of this work is on the use of explainability as a debugging tool rather than on achieving state-of-the-art driving performance, a simpler architecture facilitates the analysis and interpretation of the model behavior.

4.3 Training

The model was implemented and trained using the PyTorch framework. Training was carried out for 20 epochs in all configurations. The loss function is a weighted combination of the Mean Squared Error (MSE) for both the steering and throttle outputs:

$$\mathcal{L} = w_s \, \mathcal{L}_{\text{steering}} + w_t \, \mathcal{L}_{\text{throttle}} \tag{3}$$

where w_s and w_t are the weights assigned to each component. The model was optimized using the Adam optimizer. In addition to the MSE training loss, the Mean Absolute Error (MAE) for both steering and throttle was monitored during training to track the magnitude of the prediction error in interpretable units. Fig. 3 shows the training curves for the basic configuration. This configuration is named "One weather condition" as the complete raw RGB images feed the model input and the dataset has been recorded only in daylight in CARLA.

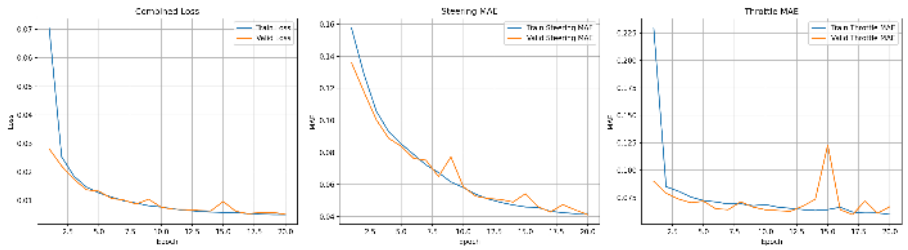


Fig. 3. MSE and MAE training curves for the "One climate" configuration without image cropping.

We performed an online evaluation of this trained model driving the car with it as the car controller, ten times, at Town02 of CARLA as the test circuit, and measuring some performance metrics as shown in Table 1. This configuration is used as the baseline for comparison with the models described at section 5. The performance of the "One weather condition" model is good-enough as in all the tests the car successfully finishes the desired lap.

5 Advantages of using AI-explainability

After the baseline model successfully completed the laps in the tests, we use explainability as a debugging tool to analyze and improve the behavior of our trained models. With SHAP and Grad-Cam we generate visual explanations to inspect which image regions drive the predictions of the model, for instance to evaluate the effect of cropping strategies, and to investigate failure cases such as those caused by shadows or spurious objects. The results of this analysis inform concrete design decisions regarding input preprocessing and training dataset composition.

5.1 Cropping

As seen in Fig. 4, the model of "One weather condition" configuration sometimes pays attention to upper parts of the input image, which may be not so relevant for making driving decisions. Cropping the input images, both for training and for inference time, may help to focus all computing power of the model in the most informative parts of the image when used for driving. And this may improve the performance of the model. To explore this hypothesis we cropped the top 45% rows of the input images and measured the performance of this configuration, named "One weather condition, with crop".

Fig. 4. Two input images, not cropped, and the focus of attention of the model on them according to SHAP.

5.2 Shadows

As seen in the middle row of Fig. 5, the model of "One weather condition" configuration tends to ignore shadow areas and pays attention to pavement areas

with daylight, missing the limits of the pavement, which use to be very informative for driving decisions. In order to make the model learn that those limits are important to pay attention to we recorded a new training dataset with the CARLA scenario in a second weather condition. This way the dataset included images from the same areas but with different pavement lightning conditions, shadow sizes and orientations. To explore this hypothesis we built a combined dataset from both lightning conditions and measured the performance of this configuration, named "Two weather conditions, with crop".

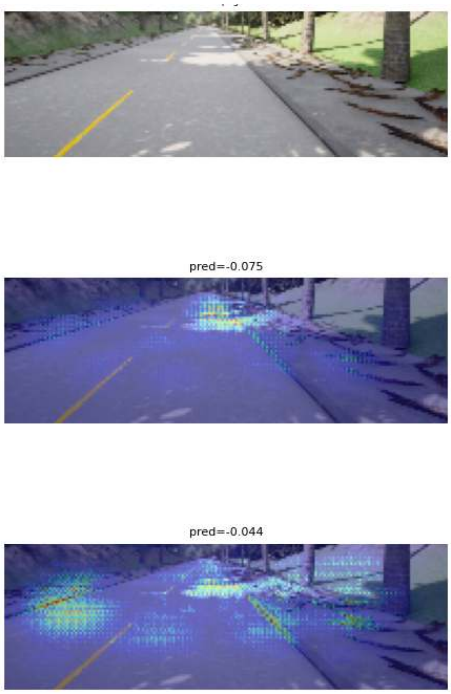


Fig. 5. Initial images (top); focus of attention in regular dataset (middle row); and training with Two-weather-conditions dataset (bottom row).

5.3 Evaluation

Three configurations were compared, which differ in the input preprocessing and the diversity of the training data:

- **One weather condition, without crop**: trained with data from a single weather condition using the full camera image as input.
- **One weather condition, with crop**: trained with data from a single weather condition, applying a crop to the input image to remove the upper portion (sky and distant scenery).

– **Two weather conditions, with crop**: trained with data from two different weather conditions, also with cropped input images.

The trained models were evaluated on CARLA Town02, a map not seen during training, to assess their generalization capability. Table 1 summarizes the results. The performance metrics used are: number of successful runs, out of 10; failure rate (collisions, lane invasions...); and lap time.

Table 1. Performance comparison of the three training configurations.

Configuration	Completion	Failure rate	Time (s)	Notes
Two weather conditions, with crop	100%	20%	25.77	Smooth driving
One weather condition, with crop	100%	30%	20.73	Oscillates
One weather condition	100%	100%	18.70	Wrong lane

All three configurations achieve a 100% completion rate, meaning the vehicle completes the circuit in all attempts. However, they differ substantially in driving quality. The configuration without cropping fails to follow the correct lane in all laps, indicating that the model attends to irrelevant image regions. Introducing the crop significantly reduces the failure rate, and adding a second weather condition to the training data further improves performance, yielding smooth driving with only a 20% lap failure rate. These results already suggest that both input, preprocessing and training data diversity, play a critical role in the model behavior when driving and its performance.

6 Conclusions

In this paper we have presented two preliminary examples of using XAI tools, SHAP and Grad-Cam, to debug and improve the performance of a Deep Learning end-to-end visual control model for the visual follow-lane application. They provide insights about which parts of the input image attract the network attention in its decision making.

First example suggested to crop input images under the horizon line to prevent distractors in the upper part of the image. Distractors were discovered in the explainability analysis with SHAP. The model trained with cropped images outperforms the basic model with complete images.

In the second example, an explainability analysis was performed specifically in the images from the testing dataset where the trained model and the supervised output were significantly different. It probed that the model properly focuses on lane edges in general, but in some of those cases also on shadows over the pavement. New images were collected in different weather and lightning conditions to show the model more training data, even of same places, with shadows in different orientations and without any shadows. Just to let it learn that shadows are not important for decision making. The network trained with this extended dataset showed less distractions.

These two useful examples show a way about how the explainability tools may help when developing AI powered robotics solutions.

As future lines we plan to test this approach to debug AI-based applications with real robots, such as the AWS DeepRacer. In addition, we intend to explore also more powerful Vision Language models (VLMs) that no only deliver continuous numerical actions but also a text explanation of them.

Acknowledgments. This work was supported by Spanish Agencies AEI and CDTI as part of the Ministry of Science, Innovation and Universities with grant number PLEC2023-010303 (GAIA); and by the project iRoboCity2030-CM (Ref TEC-2024/TEC-62), from Programa de Actividades de I+D entre grupos de investigación de la Comunidad de Madrid en Tecnologías 2024.

References

1. Atakishiyev, S., Salameh, M., Yao, H., Goebel, R.: Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions. arXiv preprint arXiv:2112.11561 (2021)
2. Bojarski, M., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., Muller, U., Zieba, K.: VisualBackProp: Efficient visualization of CNNs for autonomous driving. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 4840–4847. IEEE (2018). https://doi.org/10.1109/ICRA.2018.8461053
3. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, Bernhard graves, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
4. Bojarski, M., Yeres, P., Choromanska, A., Choromanski, Krzysztof graves, B., Jackel, L.D., Muller, U.: Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv preprint arXiv:1704.07911 (2017)
5. Cultrera, L., Seidenari, L., Becattini, F., Pala, P., Del Bimbo, A.: Explaining autonomous driving with visual attention and end-to-end trainable region proposals. Journal of Ambient Intelligence and Humanized Computing **14**, 3265–3278 (2023). https://doi.org/10.1007/s12652-023-04550-8
6. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
7. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)
8. El Houd, A., El Hachem, C., Painvin, L.: Deep learning model explainability for inspection accuracy improvement in the automotive industry. In: Proceedings of the IEEE International Conference on Industrial Informatics (INDIN). pp. 1–6. IEEE (2021)
9. Flora, M., Potvin, C., McGovern, A., Handler, S.: Comparing explanation methods for traditional machine learning models part 1: An overview of current methods and quantifying their disagreement. arXiv preprint arXiv:2211.08943 (2022)
10. Kim, J., Canny, J.: Interpretable learning for self-driving cars by visualizing causal attention. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2961–2970 (2017)

11. Kim, J., Rohrbach, A., Darrell, T., Canny, J., Akata, Z.: Textual explanations for self-driving vehicles. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 577–593 (2018)
12. Kuznietsov, Y., Gittens, A., Iagnemma, K.: Explainable AI for safe and trustworthy autonomous driving: A systematic review. IEEE Transactions on Intelligent Transportation Systems **25**(12), 18243–18258 (2024). https://doi.org/10.1109/TITS.2024.3474469
13. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3), 31–57 (2018). https://doi.org/10.1145/3236386.3241340
14. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 30, pp. 4765–4774 (2017)
15. Mikołajczyk-Bareła, A.: Data augmentation and explainability for bias discovery and mitigation in deep learning. arXiv preprint arXiv:2308.09464 (2023)
16. Molnar, C.: Interpretable Machine Learning. Christoph Molnar, 3 edn. (2025), https://christophm.github.io/interpretable-ml-book
17. Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B.: Definitions, methods, and applications in interpretable machine learning. Proceedings of the National Academy of Sciences **116**(44), 22071–22080 (2019). https://doi.org/10.1073/pnas.1900654116
18. Nowak, T., Nowicki, M.R., Cwian, K., Skrzypczyński, P.: How to improve object detection in a driver assistance system applying explainable deep learning. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV). pp. 226–231. IEEE (2019)
19. Paniego, S., Calvo-Palomino, R., Cañas, J.M.: Enhancing end-to-end control in autonomous driving through kinematic-infused and visual memory imitation learning. Neurocomputing **600**, 128161 (2024)
20. Paniego, S., Shinohara, E., Cañas, J.M.: Autonomous driving in traffic with end-to-end vision-based deep learning. Neurocomputing **594**, 127874 (2024)
21. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019). https://doi.org/10.1038/s42256-019-0048-x
22. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017). https://doi.org/10.1109/ICCV.2017.74
23. Shapley, L.S.: A value for n-person games. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games, vol. II, pp. 307–317. Princeton University Press (1953)
24. Zablocki, É., Ben-Younes, H., Pérez, P., Cord, M.: Explainability of deep vision-based autonomous driving systems: Review and challenges. International Journal of Computer Vision **130**, 2425–2452 (2022). https://doi.org/10.1007/s11263-022-01657-x
25. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016). https://doi.org/10.1109/CVPR.2016.319

Towards autonomous navigation for small indoor drones

Alfonso Núñez Ferreiro[3][0009-0009-8545-838X], Izán Fernández Moreno[2][0009-0008-2212-7287], Xose R. Fdez-Vidal[1][0000-0001-9388-7461], and Roberto Iglesias Rodríguez[1][0000-0002-6279-5190]

[1] Centro de Investigación en Tecnologías Inteligentes (CiTIUS)
[2] Escola Politécnica Superior de Enxeñería (EPSE)
[3] Tecnoloxías da Información, Escola Politécnica Superior de Enxeñería (EPSE)

Abstract. Autonomous navigation of unmanned aerial vehicles indoors presents significant challenges due to the lack of positioning systems and the need to use lightweight, low-cost sensors. This paper investigates a hierarchical navigation architecture based exclusively on monocular vision, consisting of a reactive obstacle avoidance layer using depth estimation and a deliberative layer based on the *NoMaD* visual navigation model. The *HydraNet* architecture is used for depth estimation, and its generalization capability is evaluated using different datasets and cross-domain adaptation strategies. The results show a significant loss in performance when applying the models outside the training domain, although the use of a mixed dataset improves robustness and the balance between accuracy and generalization, achieving a mean absolute error (MAE) of 0.341 and a δ_1 value of 0.829 on the *NYUv2* dataset. Furthermore, *NoMaD* is evaluated using an angular error metric on sequences from the *TUM RGB-D* dataset. The model achieves a median angular error of 35.73° and a mean error of 65.44°, indicating adequate performance on a significant portion of the samples, although with high variability associated with extreme errors.
The results obtained demonstrate the value of combining *HydraNet* and *NoMaD* in an end-to-end architecture designed to mitigate the hardware limitations of lightweight drones. They also highlight the need to continue improving the generalizability and robustness of the models to ensure their effective application in real-world environments.

Keywords: Autonomous Navigation · Indoor UAVs · Monocular Vision · Monocular Depth Estimation · Visual Navigation · Obstacle Avoidance · Hierarchical Navigation · Deep Learning

1 Introduction

Unmanned aerial vehicles (UAVs) have become increasingly important in applications such as inspection, surveillance, logistics, and autonomous exploration. However, autonomous navigation in indoor environments remains a challenge due

to the absence of GNSS signals and the need to detect and avoid obstacles in real time[1, 6]. Traditionally, autonomous indoor navigation has been addressed using simultaneous localization and mapping (SLAM) [16] techniques and sensor-based systems such as LiDAR or RGB-D cameras. Among the most influential visual approaches is ORB-SLAM2 [7], widely used as a benchmark in navigation and visual reconstruction. However, these systems typically require an accurate estimate of the environment's geometry and, in many cases, additional sensors that increase the platform's cost, weight, and energy consumption.

In this context, monocular vision emerges as an attractive alternative due to its low cost and ease of integration [3]. However, estimating the three-dimensional structure of the environment from a single camera remains a complex problem. Recent advances in deep learning have significantly improved monocular depth estimation [10], while goal-driven visual navigation models, such as *NoMaD*, have demonstrated their capability to generate trajectories from visual information without the need for explicit maps [12].

In lightweight aerial platforms, reducing weight and energy consumption is particularly important for improving flight endurance and minimizing risk during operations in confined spaces [2].

To evaluate the feasibility of this approach, we analyze separately the performance and generalization capabilities of the models employed at each level of the architecture. In particular, we study monocular depth estimation using *HydraNet* with different datasets and cross-domain adaptation strategies, as well as *NoMaD*'s capability to predict navigation directions based on visual information.

The experimental results show that generalization capability is one of the main challenges facing monocular vision-based systems. In the case of depth estimation, cross-validation results reveal a significant degradation in performance when models are applied outside the training domain. However, training with a mixed dataset yielded the best trade-off between accuracy and robustness, achieving an MAE of 0.341 and a δ_1 value of 0.829 on *NYUv2*. Meanwhile, the evaluation of *NoMaD* showed a median angular error of 35.73°, indicating adequate performance for a significant proportion of the samples, although the presence of outliers raised the mean error to 65.44°. These results highlight both the potential and the current limitations of combining monocular depth estimation and visual navigation for autonomous indoor UAV navigation.

Our contributions are threefold:

1. We propose a hybrid autonomous navigation architecture for indoor UAVs that integrates monocular depth estimation for reactive obstacle avoidance with a goal-based visual navigation model for high-level decision-making.
2. We evaluate the generalization capability of *HydraNet* for monocular depth estimation, including knowledge distillation techniques.
3. We conduct a detailed analysis of the behavior of *NoMaD*, studying the distribution of its prediction errors.

2 Method

The selected models are characterized by their low computational cost, which allows them to run in real time on platforms with limited resources, such as drones. This feature is essential for ensuring responsive and safe autonomous navigation without the need for high-performance hardware.

2.1 Datasets

Two different *datasets* are used to train the depth model: *NYUv2* [8], which is a publicly available dataset captured with a *Microsoft Kinect* camera, and *Campus*, captured by one of the authors of this article. The *CAMPUS* dataset consists of 2450 framesets, each comprising an RGB image and its corresponding depth map (ground truth). The images were captured using an Intel RealSense D435i camera in various indoor spaces at the Higher Polytechnic School of Engineering of the University of Santiago de Compostela (Lugo Campus). The dataset was divided into three mutually exclusive subsets: training (1715 framesets, 70%), validation (368 framesets, 15%), and test (367 framesets, 15%). This partition remained fixed throughout all experiments.

To evaluate NoMaD, we used the *TUM RGB-D* evaluation *dataset* [13], specifically the sequences belonging to the *Robot SLAM* category.

2.2 Low level: reactive navigation using monocular depth estimation with HydraNet

Depth estimation is performed using the *HydraNet* model, a convolutional architecture consisting of an *encoder* based on *MobileNet V2* [11] and a *decoder* based on *RefineNet* [5]. The original architecture incorporates two output heads: one for semantic segmentation and the other for depth estimation. In this work, only the depth head is used, with the goal of obtaining a metric depth map that reveals the depth associated with each pixel in the image relative to the camera's reference system. This functionality is essential for enabling autonomous drone navigation.

Thus, given an RGB image, *HydraNet* generates a depth map that makes it possible to determine whether there are any obstacles close enough to pose a danger to the drone and, if so, to avoid them in time to prevent a collision.

The following metrics are used to evaluate the model's performance:

- Errors:
  - **MAE (Mean Absolute Error): MAE** $= \frac{1}{N}\sum |y_i - \hat{y}_i|$
  - **RMSE (Root Mean Squared Error): RMSE** $= \sqrt{\frac{1}{N}\sum (y_i - \hat{y}_i)^2}$
  - **AbsRel (Absolute Relative Error): AbsRel** $= \frac{1}{N}\sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i}$
  - **SqRel (Squared Relative Error): SqRel** $= \frac{1}{N}\sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{y_i}$
- Accuracy Metrics:

- **Metric δ_1:** a metric that quantifies the proportion of pixels whose predictions have a relative error below the threshold (1.25), providing a measure of the model's performance in terms of prediction accuracy.

After training, cross-validation is performed (using the metrics mentioned earlier) to test the model's generalization capability. That is, the model is trained on the *NYUv2* dataset and evaluated on *Campus*, and vice versa.

To improve the model's generalization capability, we propose implementing three strategies: *Fine-tuning*, *Continuous Learning (Rehearsal)*, and *Knowledge Distillation*.

Ultimately, the *Knowledge Distillation* strategy was chosen because it is a relatively simple technique to implement and was well-suited to the specific case at hand—a model whose performance varied significantly when applied to two specific domains and which needed to improve its generalization capability.

This technique uses pre-trained models as teachers to guide the learning of a student model [4]. The loss function combines the error relative to the true value with the discrepancy between the student's predictions and those of the teachers specialized in each domain. In this way, knowledge from multiple domains is transferred to a single model, improving its capability to generalize. However, it may inherit biases from the teacher models and increase computational cost.

2.3 High level: navigation using NoMaD

NoMaD [12] consists of two components: a prediction model and a navigation strategy.

The model takes as input a sequence of visual observations and a target image, producing two outputs: an estimate of the time distance to the target along a reference trajectory, and a normalized relative displacement in the (X) and (Y) axes indicating the direction of motion.

Navigation relies on a previously recorded reference trajectory stored as a sequence of images. During execution, the agent compares its observations with the trajectory images to locate the closest target position, then follows the predicted displacement toward the next waypoint, repeating the process until reaching the destination.

NoMaD Evaluation Since *NoMaD* is trained with normalized displacements, its predictions do not correspond to physical quantities, preventing direct evaluation using absolute metric errors (e.g., meters).

Nevertheless, the motion direction is preserved despite the scale difference. Therefore, evaluation focuses on the angular displacement prediction, measuring the model's ability to estimate the direction of relative motion between the observation and the target image.

The predicted displacement angle is computed as:

$$\theta = \text{atan2}(y, x)$$

where x and y are the relative displacement components.

The predicted and actual angles are compared using an angular error that accounts for angle periodicity. Rather than direct subtraction, the unit-circle formulation is used:

$$e = |\text{atan2}\left(\sin(\theta_{real} - \theta_{pred}), \cos(\theta_{real} - \theta_{pred})\right)|$$

This ensures the angular error is always the minimum difference between two directions, avoiding discontinuities at the $\pi/-\pi$ boundary.

2.4 Hardware

A laptop with the following specifications was used to train the network:

- Graphics card: RTX 4060 with 8 GB of RAM.
- Processor: Intel® Core™ i7
- RAM: 32.0 GB

3 Results

3.1 Depth Estimation Results

Cross-domain evaluation In order to analyze the models' generalization capability, a cross-validation was performed between the *NYUv2* and *Campus* datasets.

Table 1. Cross-validation results.

Model	**Evaluation dataset**	**MAE**	**RMSE**	**AbsRel**	δ_1
Campus	Campus	0.831	1.610	0.230	0.688
Campus	NYUv2	1.077	1.373	0.464	0.361
NYUv2	NYUv2	**0.481**	**0.628**	**0.199**	**0.699**
NYUv2	Campus	2.714	4.301	0.526	0.186

The results in Table 1 show a significant decline in performance when each model is evaluated outside the domain for which it was trained.

Training with the Mixed Dataset To improve generalization performance, the Mixed dataset was created, consisting of 50% samples from *NYUv2* and 50% samples from *Campus*.

Table 2 shows that joint training resulted in balanced performance across both domains, maintaining good performance on *Campus* and significantly improving results on *NYUv2*.

Table 2. Results of the model trained on the Mixed dataset.

Evaluation dataset	**MAE**	**RMSE**	**AbsRel**	δ_1
Mixed	0.647	1.099	0.212	0.713
Campus	0.742	1.488	0.212	0.755
NYUv2	**0.341**	**0.457**	**0.139**	**0.829**

Distillation of Knowledge Knowledge distillation was performed using models trained on the *NYUv2* and *Campus* datasets as teachers, and training the student model on the Mixed dataset.

The results are shown in Table 3.

Table 3. Results from the distilled model.

Evaluation dataset	**MAE**	**RMSE**	**AbsRel**	δ_1
Mixed	0.691	1.153	0.212	0.695
Campus	0.835	1.620	0.220	0.709
NYUv2	**0.381**	**0.504**	**0.154**	**0.789**

Table 4. Evaluation on the Campus dataset.

Model	**MAE**	**RMSE**	**AbsRel**	δ_1
Campus	**0.831**	**1.610**	**0.230**	0.688
NYUv2	2.714	4.301	0.526	0.186
Distillation	0.835	1.620	0.220	**0.709**

Final Model Comparison In Tables 4 and 5, the results show that the distilled model achieves the best balance between the two domains, performing comparably to the *Campus* model on *Campus* and outperforming the *NYUv2* model on *NYUv2*.

Qualitative Comparison In Figure 1, we can see a qualitative comparison between the input RGB image, the prediction, and the ground truth.

3.2 NoMaD Results

Table 6 presents the results obtained by the *NoMaD* model. To quantify the model's performance, we used the angular error between the direction predicted by *NoMaD* and the reference direction associated with each sample.

Table 5. Evaluation on the NYUv2 dataset.

Model	**MAE**	**RMSE**	**AbsRel**	δ_1
Campus	1.335	1.677	0.604	0.319
NYUv2	0.481	0.628	0.199	0.699
Distillation	**0.381**	**0.504**	**0.154**	**0.789**

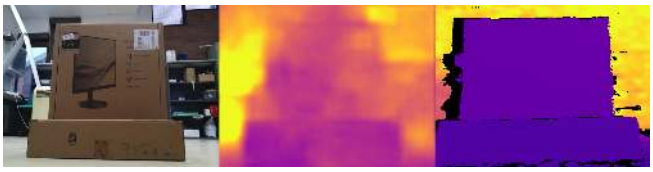


Fig. 1. From left to right: Input RGB, Predicted Depth, Ground Truth

The results show a mean angular error of 65.44°, with a standard deviation of 62.81°. Furthermore, the 95% confidence interval lies between 64.24° and 66.62°, indicating a stable estimate of the mean.

Figure 2 shows the cumulative distribution of the angular error of *NoMaD*.

Figure 3 shows a histogram of the angular error of *NoMaD*.

Table 7 summarizes several representative percentiles.

The lower percentiles indicate relatively small prediction errors. The median angular error is 35.73°, while the 25th percentile is 12.12°.

However, for the upper percentiles of the distribution, both the 90th and 95th percentiles show high errors. This behavior suggests that accuracy is concentrated primarily in typical cases, while the most extreme errors continue to represent a limitation.

These extreme errors occur mainly when there is little forward motion between images, which makes it difficult to determine the direction precisely, or in cases where there is little overlap between images.

4 Discussion

4.1 Discussion of Depth Estimation

Cross-validation reveals a significant performance loss when models are applied outside their training domain. The model trained on *Campus* performs substantially worse on *NYUv2*, while the *NYUv2* model suffers an even larger decline on *Campus*. These results confirm the limited generalization of single-domain models, which learn environment-specific characteristics (e.g., lighting conditions or geometric bias) instead of universal physical principles.

Table 6. Comparison of the average angular error.

Model	**Mean (°)**	**Sta. Dev. (°)**	**95% CI (°)**
NoMaD	65.44	62.81	[64.24, 66.62]

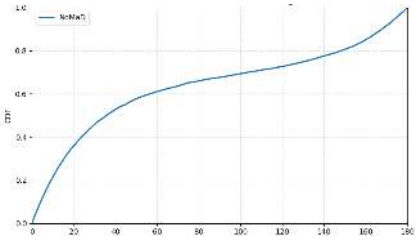


Fig. 2. Cumulative distribution of the angular error between 0° and 180°

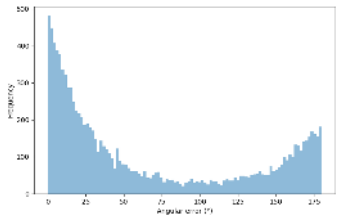


Fig. 3. Histogram of the angular error between 0° and 180°

Table 7. Percentiles of the angular error distribution

Model	P25	P50	P75	P90	P95
NoMaD	12.12°	35.73°	130.65°	167.83°	174.28°

This domain shift highlights the need for more robust cross-domain representations [14]. Training on the Mixed dataset improves robustness by encouraging domain-invariant feature learning [14]. By combining both datasets, the model learns more diverse representations, reducing dependence on environmental cues, achieving the best overall performance, and providing a strong generalization baseline.

Knowledge distillation also produces a balanced model across domains. Although it does not outperform direct training on the Mixed dataset, it preserves high fidelity (comparable to the *Campus* model on *Campus*) while greatly improving performance on the challenging *NYUv2* cross-domain scenario. This indicates that distillation acts as an effective regularization technique, transferring robust knowledge from multiple teachers to stabilize the student model's decision boundaries.

4.2 Discussion of NoMaD

The descriptive statistics reveal a highly asymmetric distribution of angular errors. Although many predictions have low or moderate errors, a considerable number of extreme errors increase both the mean and the dispersion.

The gap between the mean (65.44°) and median (35.73°) reflects this asymmetry. While half of the samples have errors below approximately 36°, the mean is skewed by extreme values. This is consistent with the high standard deviation (62.81°), indicating considerable variability. Thus, at least 50% of the predictions have moderate errors, whereas the high mean results from relatively few extreme cases.

The high standard deviation (62.81°) and P95 (174.28°) further highlight this instability, suggesting that the main limitation is not average performance but vulnerability to limited motion or image overlap. The largest errors occur when:

– Motion is small (low displacement), introducing rotational ambiguity.
– Geometric information is insufficient due to limited image overlap.

A major limitation of this study is that both architecture levels were evaluated independently rather than integrated into a real UAV platform.

5 Conclusions

This paper investigates a two-level autonomous drone navigation architecture combining monocular depth estimation for reactive obstacle avoidance and *NoMaD* for visual trajectory tracking.

Depth estimation models show limited cross-domain generalization. However, training on the Mixed dataset provides the best balance between accuracy and robustness, outperforming specialized models overall. Knowledge distillation also achieves competitive performance but does not surpass joint training.

Regarding *NoMaD*, the model correctly predicts motion direction in many samples, but large angular errors in some cases produce high variability and reduce reliability.

Overall, the results demonstrate the potential of deep learning for autonomous navigation while highlighting the need to improve cross-domain generalization and prediction robustness before deployment in real-world environments.

5.1 Future work.

The depth estimation model provides a strong baseline, exposing the generalization limits of monocular vision and motivating more advanced approaches such as *UniDepth v2* [9] and *DepthAnything v2* [15].

Future work could also develop a dedicated navigation model with similar accuracy, a smaller size, and the ability to estimate actual geometric distances in meters.

Acknowledgments. This research was supported by PID2023-153341OB-I00 funded by MCIU /AEI/10.13039/501100011033 / FEDER, EU, and it was also funded by the European Commission through the POCTEP-INTERREG program [Grant 0054_AERO-GANP_1_E]

References

1. Chang, Y., Cheng, Y., Manzoor, U., Murray, J.: A review of uav autonomous navigation in gps-denied environments. Robotics and Autonomous Systems **170**, 104533 (2023). https://doi.org/https://doi.org/10.1016/j.robot.2023.104533, https://www.sciencedirect.com/science/article/pii/S0921889023001720
2. Golodetz, S., Vankadari, M., Everitt, A., Shin, S., Markham, A., Trigoni, N.: Real-time hybrid mapping of populated indoor scenes using a low-cost monocular uav (2022), https://arxiv.org/abs/2203.02453

3. Herrera-Granda, E.P., Torres-Cantero, J.C., Peluffo-Ordóñez, D.H.: Monocular visual slam, visual odometry, and structure from motion methods applied to 3d reconstruction: A comprehensive survey. Heliyon **10**(18), e37356 (2024). https://doi.org/https://doi.org/10.1016/j.heliyon.2024.e37356, https://www.sciencedirect.com/science/article/pii/S2405844024133877
4. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015), https://arxiv.org/abs/1503.02531
5. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation (2016), https://arxiv.org/abs/1611.06612
6. Marshall, J.A., Sun, W., L'Afflitto, A.: A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems. Annual Reviews in Control **52**, 390–427 (2021). https://doi.org/https://doi.org/10.1016/j.arcontrol.2021.10.013, https://www.sciencedirect.com/science/article/pii/S1367578821000882
7. Mur-Artal, R., Tardos, J.D.: Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Transactions on Robotics **33**(5), 1255–1262 (Oct 2017). https://doi.org/10.1109/tro.2017.2705103, http://dx.doi.org/10.1109/TRO.2017.2705103
8. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
9. Piccinelli, L., Sakaridis, C., Yang, Y.H., Segu, M., Li, S., Abbeloos, W., Van Gool, L.: Unidepthv2: Universal monocular metric depth estimation made simpler. IEEE Transactions on Pattern Analysis and Machine Intelligence **48**(3), 2354–2367 (Mar 2026). https://doi.org/10.1109/tpami.2025.3628473, http://dx.doi.org/10.1109/TPAMI.2025.3628473
10. Rajapaksha, U., Sohel, F., Laga, H., Diepeveen, D., Bennamoun, M.: Deep learning-based depth estimation methods from monocular image and videos: A comprehensive survey. ACM Comput. Surv. **56**(12) (Oct 2024). https://doi.org/10.1145/3677327, https://doi.org/10.1145/3677327
11. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks (2019), https://arxiv.org/abs/1801.04381
12. Sridhar, A., Shah, D., Glossop, C., Levine, S.: Nomad: Goal masked diffusion policies for navigation and exploration (2023), https://arxiv.org/abs/2310.07896
13. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: Proc. of the International Conference on Intelligent Robot Systems (IROS) (Oct 2012)
14. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312**, 135–153 (2018). https://doi.org/https://doi.org/10.1016/j.neucom.2018.05.083, https://www.sciencedirect.com/science/article/pii/S0925231218306684
15. Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., Zhao, H.: Depth anything v2 (2024), https://arxiv.org/abs/2406.09414
16. Younes, G., Asmar, D., Shammas, E., Zelek, J.: Keyframe-based monocular slam: design, survey, and future directions. Robotics and Autonomous Systems **98**, 67–88 (2017). https://doi.org/https://doi.org/10.1016/j.robot.2017.09.010, https://www.sciencedirect.com/science/article/pii/S0921889017300647