Received 11 August 2024, accepted 5 September 2024, date of publication 10 September 2024,
date of current version 20 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3456914

RESEARCH ARTICLE

ARCog-NET: An Aerial Robot Cognitive Network Architecture for Swarm Applications Development

GABRYEL S. RAMOS[1], **FELIPE DA R. HENRIQUES**[1], **(Member, IEEE),**
DIEGO B. HADDAD[1], **(Member, IEEE),**
FABIO A. A. ANDRADE[2,3], **(Senior Member, IEEE),**
AND MILENA F. PINTO[1], **(Member, IEEE)**
[1]Federal Center for Technological Education Celso Suckow da Fonseca (CEFET-RJ), Rio de Janeiro 20271-110, Brazil
[2]Department of Microsystems, Faculty of Technology, Natural Sciences and Maritime Sciences, University of South-Eastern Norway (USN), 3184 Borre, Norway
[3]Drones and Autonomous Systems, NORCE Norwegian Research Centre, 9294 Tromsø, Norway

Corresponding author: Fabio A. A. Andrade (fabio@ieee.org)

This work was supported in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil (CAPES) under Grant 001 and in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ).

ABSTRACT This work presents an advanced cognitive architecture for networked aerial robots to implement autonomous swarm systems effectively. It focuses on designing, implementing, and evaluating an architecture that enables unmanned aerial vehicles (UAVs) to coordinate and cooperate for complex tasks, with or without human intervention. Inspired by artificial intelligence, cognitive science, and robotics, the architecture integrates perception, planning, decision-making, and adaptive learning to optimize swarm behavior in dynamic environments. The architecture uses a distributed processing model based on the ''edge-fog-cloud'' (EFC) concept. Edge-level robots handle real-time data collection, local decision-making, and environmental perception. Fog-level vehicles manage intermediate processing and supervision of the groups, while cloud servers perform comprehensive data analysis and long-term storage, being the higher-level hierarchy of the framework. This structure allows efficient distribution of computational tasks, with critical decisions made at the robot level and complex analysis done in the fog or cloud. The implementation of ARCog-NET involves deploying a multi-agent simulation system using the Robot Operating System (ROS) and Gazebo simulator, facilitating the integration of sensors, communication protocols, and data processing algorithms. The performance evaluation demonstrates the architecture's effectiveness in a wind farm inspection scenario, where the UAV swarm exhibits improved trajectory planning, collision avoidance, and data processing efficiency. Simulation results show that ARCog-NET reduces latency, increases data throughput, and enhances operational effectiveness, providing a robust platform for future developers to focus on applications and direct robot control methods.

INDEX TERMS Cognitive architecture, edge-fog-cloud technologies, swarm systems, distributed data processing, autonomy of aerial robots.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are one of the most versatile robot platforms due to their wide range of applications [1], [2]. To cite some uses, there is surveillance, naval operations assistance, search and rescue, inspection, package delivering, environmental monitoring, topography, among others [3], [4], [5], [6], [7], [8], [9]. The UAV systems also have improved their capacities when combined in groups or swarms, where coordinated vehicles can share the same tasks or individually execute parts of a bigger mission, gaining efficiency or even being able to accomplish operations where a single robot wouldn't be enough [10].

The associate editor coordinating the review of this manuscript and approving it for publication was Hadi Tabatabaee Malazi.

The integration of complex algorithms within aircraft systems has led to the development of highly specialized solutions tailored to specific applications [11]. Although many of these systems excel in improving particular functions, only a limited few contribute to advancing cognitive capabilities, particularly in decision-making processes critical to aircraft operations. Cognitive architecture plays a pivotal role in organizing data and information flow within an agent, thereby facilitating intelligent behavior emergence. However, existing research primarily focuses on low-level data interfaces like sensors and actuators, neglecting the broader abstraction of hardware [12].

Still, according to Pinto et al. [12], designing a comprehensive architecture capable of accommodating various cognitive aspects remains a critical challenge. Key areas such as knowledge representation, decision-making, and robotics demand practical solutions to embed artificial cognition effectively and decision-centric mission objectives. Achieving autonomous operation in robotic systems necessitates the integration of specific, interconnected components. A well-defined organizational structure is crucial for fostering intelligent behavior emergence and facilitating interoperation. Cognitive architectures offer promise in addressing challenges like adaptability and scalability [13], [14]

The concept of cognition in computing refers to integrating cognitive capabilities into computer systems, enabling them to understand, learn from, and respond to their environment in a manner akin to human cognition [15], [16]. This approach leverages artificial intelligence and machine learning algorithms to process and analyze vast amounts of data, make decisions, and solve problems more efficiently. The goal is to create systems that are not only reactive but also proactive and adaptive, capable of anticipating needs and offering solutions without explicit programming for every possible scenario [17], [18]. Cognition in robotics, on the other hand, focuses on imbuing robots with cognitive functions to enhance their autonomy and interaction capabilities [19], [20]. Cognitive robotics combines Artificial Intelligence (AI), Machine Learning (ML), computer vision, and natural language processing to enable robots to perceive their environment, understand context, learn from experiences, and make informed decisions. Such capabilities are essential for complex tasks like navigation, manipulation in unstructured environments, and human-robot interaction, pushing the boundaries of what robots can achieve independently [21], [22].

To cite some possible cognitive-network applications trends in robotics: *(i)* intelligent data processing (network layers can perform context-aware processing, understanding the significance of the data in relation to the robots' mission objectives); *(ii)* reduced latency for real-time decisions (it facilitates near-instantaneous decision-making by processing data in close physical proximity to the edge devices, essential for the autonomous operation of robots); *(iii)* bandwidth management and data prioritization (by intelligently analyzing data, the network - mainly fog layer - can prioritize its transit to the cloud, managing network resources efficiently); *(iv)* autonomous local storage and caching (it decides what data should be stored locally and what should be discarded or sent to the cloud, optimizing network and storage resources); *(v)* decentralized cognitive functions (the fog and edge layers enable distributed cognition, allowing a robot swarm to adapt to dynamic environments through collaborative intelligence. The fog nodes may also aid in the decentralized coordination of drones, for example, helping to manage communication, navigation, and operational algorithms among the swarm); *(vi)* enhanced resilience and redundancy (contributes to the robustness of the network by maintaining operations during cloud outages, utilizing its cognitive capabilities to manage a robot swarm independently); *(vii)* proactive security measures (cognitive network nodes, mainly in fog layer, can identify potential security threats in real-time and initiate protective protocols without waiting for cloud-based analysis); *(viii)* energy-efficient operations (distributed Edge-Fog-Cloud optimizes the energy consumption of robots by processing data locally or in intermediary network layers, reducing the need for long-range communications for every processing load).

A. MAIN CONTRIBUTIONS

The main contribution of this research work lies in the comprehensive exploration and integration of advanced technologies and methodologies to enhance the capabilities of UAV swarm networks. Specifically, this paper presents the development and implementation of Aerial Robot Cognitive Network Architecture (ARCog-NET). This cognitive architecture is a collective algorithm designed for cognitive UAV swarm networks operating within the Edge-Fog-Cloud architecture. ARCog-NET operates in a loop-oriented manner, with parallel iterations occurring at different network nodes, represented by active robots within the swarm. These nodes execute specific tasks across different levels of the network structure to ensure smooth network operation while UAVs perform their assigned missions.

The architecture encompasses various aspects such as data management, information exchange, cognitive decision-making for network and operation control, robot control actions, data processing, storage, and human interaction possibilities. Integrating fog and cloud computing resources enhances the system's mission control, supervision, and data analytics capabilities. This architecture emphasizes the importance of efficient communication strategies within the EFC cognitive UAV swarm network. It aims to improve communication among the swarm by analyzing latency, throughput, package loss, interference, and power constraints, which are crucial for achieving optimal system performance. The main contributions of this paper can be summarized as:

- **Development and implementation of a cognitive architecture designed for UAV swarm networks called ARCog-NET**

  This architecture integrates advanced technologies and methodologies to enhance the coordination and cooperation of UAVs in swarm applications. One of the standout features of ARCog-NET is its utilization of the Edge-Fog-Cloud model for distributed processing. This hierarchical structure allows for efficient distribution of computational tasks across different levels, significantly reducing latency and enabling real-time decision-making. Edge-level UAVs handle real-time data collection and local decision-making, fog-level vehicles manage intermediate processing and supervision, and cloud servers perform comprehensive data analysis and long-term storage.

- **Implementation of a cognitive decision-making process**
  The architecture incorporates cognitive decision-making processes that enable UAVs to optimize their behavior in dynamic environments. This includes data management, information exchange, robot control actions, and adaptive learning, ensuring UAVs can react swiftly to obstacles and adapt to changing conditions.
- **Development of a linear-structure trained model for image compression, implemented with Keras and already integrated into the control modules in Robotic Operating System (ROS)**
  This Deep Learning (DL) model offers a versatile and low computational cost solution for data compression. It is suitable for use with various image formats within the UAV swarm system and is a built-in asset to ARCog-NET communication features. This integration enhances the efficiency of data transmission and processing within the multi-UAV framework. The implementation and testing are detailed in Ramos et al. [23].
- **Testing the ARCog-NET in a semi-realistic software Gazebo along with ROS for evaluating the performance and functionality of the proposed approach.**
  ARCog-NET is integrated with ROS, a widely used framework for developing robotic applications. This integration enables communication and interaction between ARCog-NET and ROS-based UAV systems, facilitating the implementation and testing of cognitive algorithms within real-world robotic environments. The performance of ARCog-NET is demonstrated through a wind farm inspection scenario in software-in-the-loop (SITL) testing, showcasing its effectiveness in trajectory planning, collision avoidance, and data processing efficiency.

B. ORGANIZATION

The rest of this paper is organized as follows. Section II describes the related works and the state-of-art. Besides, it introduces the necessary characteristics for cognitive architectures and decision-making processes, which are essential for comprehending the ARCOG-net framework. Section III presents the overall proposed architecture for creating the compression model and integrating it into the multi-robot simulation and the mathematical foundations for validating the methodology. Section IV presents the simulation results and discusses the adopted algorithms. It shows the ARCog-NET being applied in a wind farm scenario. The final Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

A. COGNITIVE ASPECTS

The field of robotics has seen significant advancements, evolving from simple mechanical structures to complex autonomous systems capable of interacting with their environment in increasingly sophisticated ways. Central to this evolution is the development of cognitive capabilities that enable robots to perceive, understand, and act effectively within their surroundings [12]. According to Dominey and Warneken [24], the fields of psychology and computer science are embracing a phase of collaboration driven by advancements in robotics technology. Contrary to simulations and classic AI applications, which are limited to virtual environments, robots possess the sensory and motor skills necessary to navigate and operate within the human physical realm, enabling direct interaction.

This subsection outlines the essential aspects of cognition that a robot must possess to achieve a higher level of autonomy and interaction. These aspects include perception, learning and adaptation, decision-making and problem-solving, and memory [12], [15]. By integrating these cognitive faculties, robots can better understand their environment, make informed decisions, learn from experiences, and interact with other robots meaningfully.

Perception is the foundational cognitive aspect, enabling robots to acquire and process information in order to take action [25]. Effective perception involves not just the passive collection of data but also the active interpretation of this data to understand the context and significance of environmental stimuli. This perception data is part of the cognition process, which also involves attention, association, memory, reasoning, judgment, imagination, thought, and language [26]

For robots to operate autonomously in changing environments, they must be able to learn from experiences and adapt their behavior accordingly. ML algorithms, including supervised, unsupervised, and reinforcement learning, are crucial for this cognitive aspect [27]. Adaptive learning enables robots to improve their performance over time based on feedback from their actions and environmental changes. Autonomous robots must be capable of making decisions and solving problems in real-time to navigate complex environments and achieve their goals. This involves evaluating possible actions, predicting outcomes, and selecting the optimal course of action based on current objectives. Decision-making and problem-solving require synthesizing information from various cognitive processes, including perception and memory [12].

Note that this level of autonomy is crucial for robots operating in dynamic or unstructured environments, such as search and rescue operations and other applications [5],

[13], [28], [29]. With cognitive functions, robots can learn from their experiences and adapt their behaviors to suit their tasks and environments better. Adaptability extends the utility of robots to a broader range of tasks and conditions, improving their efficiency and effectiveness in changing scenarios. Besides, cognitive processes enable robots to identify problems, generate potential solutions, evaluate their feasibility, and implement the most suitable ones. This capability is essential for robots tasked with complex missions where predefined algorithms may not suffice, such as navigating unknown terrains or handling unexpected obstacles in manufacturing processes.

Several works from the literature proposed architectures with cognition aspects to enhance the engagement between robotic systems and their environment [12], [15], [30], [31], [32]. The Intelligent Vehicle Control Architecture (IVCA) introduced in [33] facilitates cooperation among various aerial vehicles. While IVCA is designed to be adaptable, its primary focus is on collaborative aircraft operations. It employs ontologies for its foundation but lacks details on hardware specifics or learning mechanisms. In contrast, Selecký et al. [14] developed a UAV framework emphasizing autonomous operations reliant on sensor data. Yet, it falls short in incorporating learning functionalities, and its hardware needs vary with the project scope. Aerostack [34] stands out by offering a structured system for commanding and managing autonomous, diverse teams directly from the aircraft, focusing on navigation and avoiding obstacles. Despite its advantages, Aerostack's framework has room for enhancement in Human In The Loop (HITL) integration, team collaboration, and autonomous decision-making. There is still a clear demand for frameworks incorporating cognitive abilities and effectively managing interactions between multiple robots and the HITL process. The HITL is a feature that allows the human operator to intervene, make decisions, and provide commands that guide the operations of the UAVs, particularly in scenarios where the automated system may struggle to find an optimal solution.

B. EDGE-FOG-COMPUTING NETWORK

An important aspect of the UAV multi-vehicle systems is the communication strategy between each part. The vehicles need to be able to send and receive data in a real-time pipeline to know each other's position, velocity, orientation, and flight autonomy and communicate with the Ground Control Stations (GCS) of the mission [35]. These network systems may be designed as IoT-based (Internet of Things) architectures or even be nodes of an IoT architecture itself, like fog [36], fog-cloud [37], or Edge-Fog-Cloud (EFC) networks [38]. The present work is particularly interested in EFC design due to its versatility and low exploration in robotics applications.

An EFC network represents a hierarchically structured computing framework that orchestrates data processing across multiple network layers to enhance IoT and smart application performance [39]. ''Edge'' computing allows data processing to occur at the source of data generation, effectively reducing response times and conserving bandwidth by not sending all data to the central cloud [40]. The ''fog'' layer acts as an intermediary that offers localized processing and storage, facilitating lower latency and network traffic and supporting mobility and geographical distribution [41]. Finally, the ''cloud'' provides a backbone with substantial computational power and storage for non-time-critical and complex processing tasks [42]. This tripartite model benefits from the immediacy of edge computing, the geographic distribution of fog computing, and the extensive resources of cloud computing presenting a scalable and efficient solution for modern computing challenges [43].

Edge computing is an emergent paradigm that brings computational capabilities to the network's periphery, closer to data sources like IoT devices and sensors. This approach is instrumental in reducing latency, enhancing data processing speed, and mitigating bandwidth issues typically associated with centralized cloud computing [44], [45]. By preprocessing data locally, edge computing allows for real-time analytics and swift decision-making, which are crucial in applications such as autonomous vehicles, smart cities, and healthcare monitoring systems [46], [47]. This paper assumes that robotics groups functioning as edge nodes in a network are the closest agents to the user (or operation goal). These robots are, in fact, the ones executing the mission they're applied for and dealing with this operation's events. These events can be navigation-related (a constraint for the robot continuing the trajectory) or mission-related (finding a goal or receiving a command from another robot), for example, detecting changes in environmental conditions, like temperature, humidity, or the presence of certain chemicals or gases, using onboard sensors identifying potential hazards or obstacles in the path of a drone or robot, necessitating a reroute or other immediate action, etc.

Fog computing is a distributed computing infrastructure that extends cloud capabilities to the edge of the network, enabling a new breed of applications and services [43]. By bringing the advantages of the cloud closer to where data is produced and acted upon, fog computing facilitates low-latency, network-efficient data processing, which is essential in use cases such as the Internet of Things (IoT), 5G networks, and real-time analytics [40], [48]. The decentralization inherent in fog computing allows for localized decision-making, critical in scenarios requiring immediate action, such as smart transportation systems and emergency services [42], [49].

Cloud computing has significantly influenced the field of robotics, enabling enhanced computational power, vast storage capabilities, and advanced robotics applications [50], [51]. Robots can access powerful processing resources through the cloud to perform complex tasks, collaborate in real-time, and utilize machine learning algorithms for improved autonomy and adaptability [52], [53]. This integration has given rise to the concept of ''cloud robotics'', which leverages the cloud for offloading heavy computation and providing services like mapping, navigation, and vision recognition, dramatically expanding the potential of robotic systems [54]. This paper also applies this computing concept to robotics, integrating cognition for autonomous, optimized, and adaptive decision-making and control.

C. DECISION-MAKING

Decision-making in the context of robots refers to the processes and mechanisms that allow a robot to autonomously select actions based on its perceptions of the environment, objectives, and internal state. Having a decision-maker on board allows UAVs to operate effectively in highly dynamic settings, even without centralized coordination. This capability enables UAVs to adapt seamlessly to any alterations in their surroundings. Traditional approaches to autonomous decision-making typically rely on extensive training with large datasets, which limits their ability to provide a form of intelligence that can be applied across a wide spectrum of scenarios [55].

In sectors with critical demands like the aerospace industry, it's essential for robotic systems to perform the decision-making process by exhibiting predictable and stable actions. Several techniques meet these criteria, with the most commonly employed algorithms for decision-making including Expert System (ES), Evolutionary Computation (EC), Data Mining (DM), and Case-Based Reasoning (CBR) [56]. DM is crucial for decision support systems, as it involves modeling and identifying patterns to extract knowledge effectively. Meanwhile, EC is adept at handling numerous criteria, making it ideal for addressing intricate problems in the real world. CBR is advantageous due to its minimal computational demands and strong representation skills, yet it fails to generate decisions for previously unseen scenarios. Neural Networks (NN) have shown the most remarkable outcomes, though they necessitate a substantial volume of data for training.

This paper employs a Reinforcement Learning (RL) approach for making decisions. RL is a machine learning method that teaches an agent to make optimal decisions through experiments. It focuses on the agent's interaction with an environment to achieve a specific goal without explicit instructions on how to achieve it. RL is one of the most popular research fields in the context of machine learning [57], effectively addressing various problems and challenges of artificial intelligence, as well as in the robotics field [58]. In recent years, there has been significant progress in leveraging reinforcement learning for UAV network operations and path planning. Murshed et al. [59] proposed a multi-agent deep reinforcement learning approach for weighted fair energy transfer in a UAV network, showcasing advanced coordination techniques for optimizing energy efficiency in UAV operations. Similarly, Brotee et al. [60] explored a hybrid clustering and multi-agent reinforcement learning approach for path planning in obstructed environments, highlighting the benefits of coalition operations between UAVs and UGVs.

By using RL, the robot is an agent that learns to make decisions through a system of rewards and penalties. Each action the agent performs results in feedback from the environment, which can be positive (reward) or negative (penalty). The agent uses this information to formulate a strategy that maximizes rewards over time. According to Leonetti et al. [61], managing execution and planning in real-time involves dealing with models that may not always be reliable. Besides, identifying and responding to failures may not suffice for a robot. Thus, it's essential for an intelligent agent to recognize its errors, learn from them, and minimize similar mistakes in the future. RL enables this capability, and it is the algorithm that this paper uses as a decision-making approach to the proposed framework ARCog-NET.

D. COGNITIVE ARCHITECTURE

In cognitive systems, data representation is crucial, which often manifests as a dichotomy between symbolic and sub-symbolic paradigms [62]. Symbolic representation involves direct symbols such as words and ontologies, facilitating knowledge verification and human understanding of decisions, making it particularly suitable for aircraft applications. Conversely, the sub-symbolic paradigm employs parallel data representations, like neural network weights, which are not directly accessible [12].

Generic architectures like State, Operator and Result (SOAR) [31] and Adaptive Control of Thought–Rational (ACT-R) [63] initially lack specific definitions for knowledge representation. However, subsequent works have proposed structures capable of both symbolic [31] and sub-symbolic representations [64]. For this work, symbolic representation primarily facilitates block interaction.

Another critical aspect is the system's ability to handle natural language. Extensions like Natural Language Understanding - State, Operator and Result (NL-SOAR) [65] and Chunk Hierarchy and Retrieval Structures (CHREST) [31] enhance general architectures to comprehend and generate natural language outputs, supporting both simulated and real-time environments. Our proposed architecture utilizes symbolic knowledge for block interaction, ensuring a comprehensive understanding of reasoning processes.

Effective problem-solving and decision-making within dynamic environments are crucial for autonomous systems. Learning from previous experiences enhances the system's adaptability and reduces programming workload. Paper [65] reviews decision-making methodologies in SOAR and ACT-R, utilizing expert knowledge and situational memory combined with rule-based logic. Similar approaches are seen in [14] and Sanchez-Lopez et al. [34], incorporating heuristic methods and centralized activity management for task distribution and path planning. While some implementations, such as Sampedro et al. [66], focus on specific tasks, broader and general mission decision-making capabilities remain underdeveloped in current research. Our work addresses this gap by integrating comprehensive decision-making mechanisms suitable for diverse mission objectives.

In aerial applications, consistent and predictable algorithms are essential for proper verification and validation. Aerial Robotics Cognitive Architecture (ARCog), proposed by Pinto et al. [12], presents an architecture focused on HITL and team coordination for autonomous operations. The approach relied on language for information representation and decision-making processes. ARCog incorporates a functional block utilizing the Case-based Reasoning approach to learn and make decisions. Its architecture is designed for compatibility with Linux-based hardware, enabling as close to real-time data processing onboard the aircraft.

E. MULTI-ROBOT COLLABORATION

Several works presented ideas for simulating multi-UAV systems and their communication network integration. Luna et al. [67] presented a multi-UAV application for area sensing with a novel path planning algorithm, with both simulation and field test robust results. However, besides presenting the network architecture and communication protocols, one needs to consider the communication network constraints (such as latency) that influence the mapping mission.

In Alfeo et al. [68], a strategy for UAV swarm coordination is also presented based on the biomimetic behavior of insect groups. The paper has a complete approach regarding group control modeling, and the authors presented a nice literature review regarding the theme. Still, more information is needed on practical and physical conditions for those systems, hardware constraints, and communication architecture, presenting only mathematical simulations and their results.

The research in multi-UAV systems that most approaches and inspired the present work are based in Robotics Operating System (ROS) [69], with Gazebo [70] physics simulator integration. ROS is a collection of software frameworks for robot development, which provide the functionality of an operating system on a heterogeneous cluster of computers. ROS also provides standard operating system services, such as hardware abstraction, low-level device control, the implementation of commonly used functionality, message passing between processes, and package management.

Sets of running ROS processes are represented in graph architecture. The processing occurs in nodes that can receive and send messages, such as multiplex sensors, control, state, planning, actuator, etc. This paper uses the system to develop the multi-UAV control packages and send and receive data between each node (or robot). The developed ROS packages also run the autoencoder for data compression in the simulation.

The ROS also has a package named Micro Air Vehicle ROS (MAVROS) [71], which is an extendable communication node for ROS with a proxy for the Ground Control Station. This package allows the developer to build applications using Micro Air Vehicle Link (MAVLINK) [72], which is a very lightweight messaging protocol for communicating with drones and between onboard drone components. This composition provides functionality for building abstraction of controllers, sensors, and actuators for real drones with PX4 Autopilot [73] firmware, simulating physics conditions in Gazebo and generating ready-to-deploy routines in real Flight Controller Units (FCUs).

The Gazebo is an open-source 3D robotics simulator. It integrates the Open Dynamic Engine (ODE) physics engine, OpenGL rendering, and code for sensor simulation and actuator control. The Gazebo can use various high-performance physics engines like ODE, and Bullet [74]. It provides realistic rendering of environments, including high-quality lighting, shadows, and textures. It can model sensors that ''see'' the simulated environment, such as laser range finders, cameras (including fish-eye), Kinect-style sensors, etc. In the present work, the Gazebo software simulates the environment, the main sensor (an RGB camera), and the physics acting on the proposed framework.

In the work of Anagnostopoulos et al. [75], a framework combining ROS and network simulators (Artery/OMNET++ [76]) is proposed, focusing on establishing a platform for cooperative autonomous vehicles system developments. This is a significant contribution since neither ROS nor Gazebo has built-in support for complex communication simulations (covering the OSI model).

Similarly to the previous citation, Mohini [77] also presented a framework for integrating a ROS/Gazebo simulation with a benchmark network simulator NS-3 [78]. The author used the PX4 Autopilot [79], [80] packages for ROS, building a multi-UAV system simulation that can be adapted for several communication systems between nodes or robots of the group (such as Wi-Fi or GSM). The author also studied the effects of communication losses and delays, taking into account the robustness of the system modeling. The main contribution of the present work regarding this is data compression, which [77] did not cover.

Following a similar path, Acharya et al. [81] and [82] proposed similar frameworks for integrating ROS and network simulators for multi-robot systems development. Both types of research deal with time synchronization, as ROS/Gazebo simulations are based on continuous iterations, and network simulators are based on discrete event time steps. Finally, Pinto et al. [13] presented a framework for analyzing fog-cloud computing cooperation applied to information processing in UAVs. The results are based on mathematical data processing models in a multi-robot communication system and communication pipeline parameters. Still, they lack physical simulation results of real robots since the presented results rely only on mathematical modeling.

Regarding data compression, Liu et al. [83] presented an autoencoder model for compressing scientific floating-point data without losing key information in data. A comprehensive study on autoencoders was conducted, but the author's model still needs to be tuned to reach higher compression rates without losing crucial data. Grassa et al. [84] and Zebang and Sei-ichiro [85] proposed two linear autoencoder models. In [84], the model is trained with satellite multispectral and hyperspectral RGB images. The second reference used a densely connected autoencoder for data compression, which was trained with high-resolution images to get richer feature extraction, then used a U-net network to decrease the distortion caused by compression. Both models demonstrated improvements compared to other methods (such as traditional JPG) but were slow for real-time data compression, which is necessary for the setup under configuration.

Unlike the mentioned works, this work aims to develop a multi-UAV simulation framework with ROS and Gazebo packages and integrate it with a data (image) compression DL model, developed with Keras and already integrated into the control modules in ROS, evaluating its robustness within the robot system. This compressor is chosen because of its versatility and low computational cost. Once trained, the model can be used with several image formats. This research presents a generic multi-UAV development platform with reconfigurable separated sensors and controls.

Table 1 provides a comparative overview of some of the cited works alongside key characteristics relevant to our proposed approach. In this table, "Yes" means the work addresses the respective aspect, and the symbol "—" means the aspect is not applicable or not mentioned in the work.

Most cited works demonstrate some decision-making capability, albeit with different approaches and methodologies. While some explicitly address this aspect, others do not focus extensively. Integration with ROS and/or Gazebo for simulation purposes is a common theme among the works, facilitating the development and testing of robotic systems in simulated environments. However, fewer works concentrate on physical robot simulation, simulating real-world hardware components and interactions. This aspect is critical for validating algorithms and behaviors in realistic settings. Similarly, data compression techniques, although not a primary focus in many works are addressed in some ways to manage and transmit sensor data efficiently, especially in resource-constrained systems.

TABLE 1. Comparison of key aspects in related works for UAV swarm architecture.

Work	Decision-making Capability	ROS Integration	Physical Robot Simulation	Data Compression
Quigley et al. (2009)	-	Yes	-	-
Laird et al. (2012)	-	-	-	-
Ball et al. (2013)	-	-	-	-
Tweedale et al. (2014)	-	-	-	-
Selecky et al. (2015)	Yes	-	-	-
Sampedro et al. (2016)	Yes	-	-	-
Butz et al. (2016)	-	-	-	-
Sánchez-Rada et al. (2017)	Yes	-	-	-
Pinto et al. (2019)	Yes	Yes	-	-
Pinto et al. (2020)	Yes	Yes	Yes	-
Christos et al. (2021)	-	Yes	-	-
Acharya et al. (2021)	-	Yes	-	-
Calvo et al. (2021)	-	Yes	-	-
Liu et al. (2021)	-	-	-	Yes
Lunas et al. (2022)	-	-	Yes	-
Antonio et al. (2022)	-	-	Yes	-
Lagrassa et al. (2022)	-	-	-	Yes
Mohini et al. (2023)	-	Yes	-	-
Proposed Approach	Yes	Yes	Yes	Yes

III. THE PROPOSED ARCog-NET ARCHITECTURE

The basic cycle of ARCog-NET functioning is a loop-oriented collective algorithm in which each iteration happens in parallel at different network nodes (where a network node can be understood as an active robot of the swarm) that occupies different levels of the structure. Each level has specific tasks to accomplish to keep the whole network working while the robots execute the mission that the swarm is tasked with. In general, data structures, flows and filtering, information exchange, cognitive decision-making for network control and operation control, robot control actions, data processing, data storage, and human interaction possibilities for each network level that should run at each node. Figure 1 presents a basic representation of ARCog-NET with two groups of edge agents coordinated by two fog coordinators each and connected to a cloud mainframe for mission control and supervision.

Various communication issues can significantly impact the system's overall efficacy and performance in the context of the EFC cognitive UAV swarm network generated by ARCog-NET. To cite some:

- Latency: Critical for real-time operations, latency becomes a concern when data must traverse multiple layers from edge to cloud, potentially delaying decision-making processes.
- Bandwidth Limitations: The drone's high volume of data can exceed the available bandwidth, leading to transmission bottlenecks, especially during peak data transfer periods.
- Interference: Communication can be disrupted by environmental factors, electronic interference, or competing signals, leading to data loss or miscommunication.
- Security and Privacy: Open wireless communication channels present vulnerabilities to security breaches and

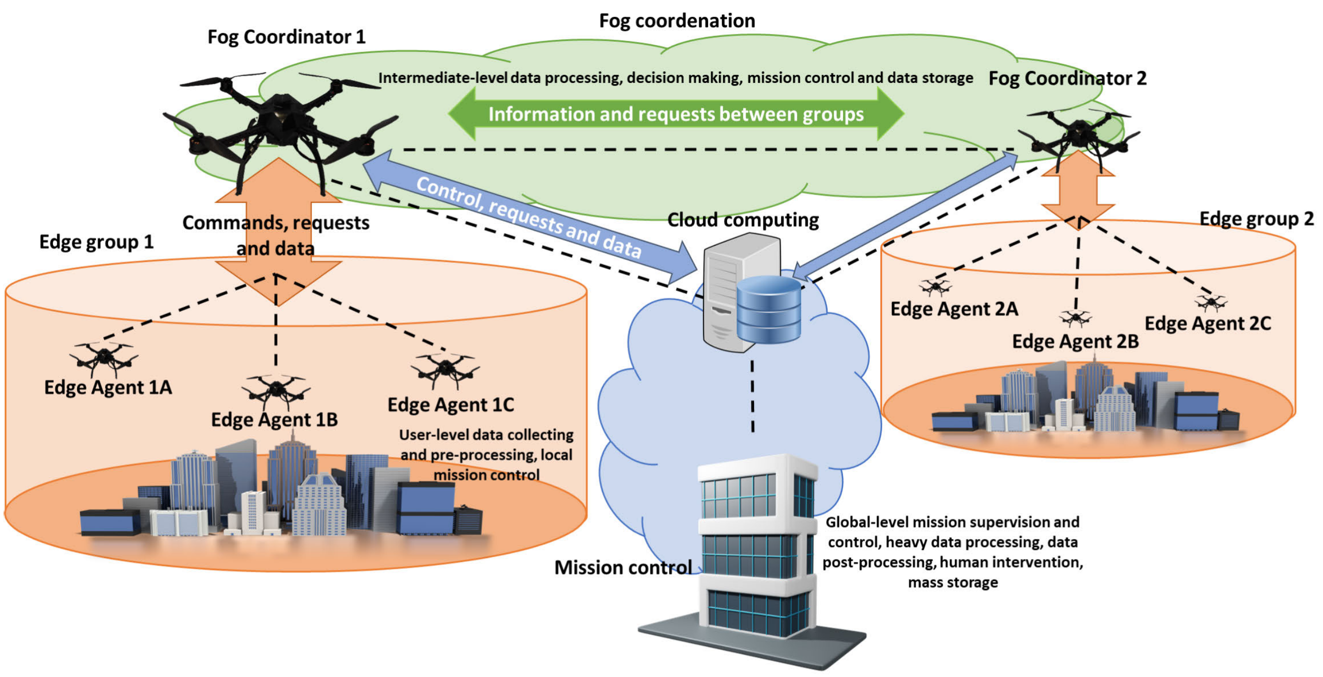

Fog coordenation
Fog Coordinator 1
Intermediate-level data processing, decision making, mission control and data storage
Fog Coordinator 2
Information and requests between groups
Control, requests and data
Commands, requests and data
Cloud computing
Edge group 1
Edge group 2
Edge Agent 2A
Edge Agent 2B
Edge Agent 2C
Edge Agent 1A
Edge Agent 1B
Edge Agent 1C
User-level data collecting and pre-processing, local mission control
Global-level mission supervision and control, heavy data processing, data post-processing, human intervention, mass storage
Mission control

FIGURE 1. ARCog-NET basic representation.

privacy intrusions, necessitating robust encryption and security protocols.

- Scalability: As the drone network expands, the communication infrastructure must scale accordingly without compromising performance or increasing latency.
- Energy Constraints: Drones are limited by battery life, with communication protocols consuming significant energy. Thus, efficient energy use is paramount for sustained operations.

Overcoming these communication challenges is crucial for deploying effective and autonomous cognitive drone swarm networks, and the proposed framework is designed to simulate those constraints, as will be discussed later. ARCog-NET was also designed with a specific message format that keeps the same structure regardless of the dataflow between layers or nodes and is always broadcast through the MAVLink protocol. This message structure is basically an array of arrays following the representation presented in Figure 2.

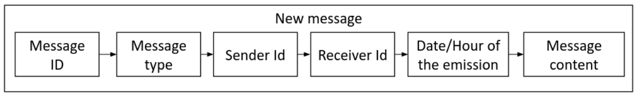


FIGURE 2. ARCog-NET message structure representation.

Message content is the core of exchanging information, passing commands through the network, or requests addressed downstream or upstream. In the present architecture, these messages were divided into categories (or workflows), which represent the kind of message (data, processed data, request, or command) and the node (agent, coordinator, or server) they're being sent or coming from. Table 2 represents these workflow descriptions, which will be important to understand the data structures of the layers in the following subsections dealing with the implemented architecture's algorithms and data. It is important to notice that future developers can add workflows to the basic structure.

TABLE 2. Complete workflow descriptions.

Workflow type	Workflow description
A1	Agent data
A2	Agent processed data
A3	Agent request to coordinator
A4	Agent request to server
C1	Coordinator data
C2	Coordinator processed data
C3	Coordinator request to agent
C4	Coordinator request to server
C5	Coordinator command to agent
S1	Server data
S2	Server processed data
S3	Server request to agent
S4	Server request to coordinator
S5	Server command to coordinator
S6	Server command to agent

The size of ARCog-NET message fields and the control overhead received per message or communication depends on the specific implementation details of the network protocol (in this case, MAVLink [72]) and the data structures used in the communication process. Every packet has 8 bytes dedicated to the MAVLink header, 2 checksum bytes (totaling 10 control overhead bytes), 32 bytes for message ID, message type, sender ID, and receiver ID, respectively (totaling 128 bytes), and 48 bytes for date/hour of message emission. The message content may vary in size. A position data message, for example, has 12 bytes (4 for each one of the x, y, and z coordinates), and a high-definition image is sent as a message occupying up to 70 bytes per packet. This way, ARCog-NET messages always have up to 256 fixed bytes.

Note that MAVLink does not handle routing or Media Access Control (MAC) layer functions directly, it can be used effectively in a networked environment with multiple nodes by leveraging additional networking protocols and infrastructure. In ARCog-NET architecture, for instance, MAVLink is used for transmitting structured messages across the network, with the actual data routing and handling managed by the underlying network infrastructure. This approach allows MAVLink to support communication among multiple UAVs and nodes in the network, facilitating coordinated operations and data exchange

A. THEORETICAL MODELING AND ALGORITHMS

The choice of employing a cognitive-based decision-making flight algorithm grounded in Particle Swarm Optimization (PSO) for Unmanned Aerial Vehicle (UAV) operations is underpinned by several compelling factors. First, cognitive-based algorithms leverage principles from cognitive science and artificial intelligence to simulate human-like decision-making processes, enhancing the autonomy and adaptability of UAVs in dynamic environments. PSO, inspired by the social behavior of birds flocking or fish schooling, is particularly well-suited for optimization problems involving multiple objectives and constraints, such as those encountered in UAV flight planning and execution. Its ability to efficiently explore and exploit the search space enables UAVs to identify optimal flight paths that balance various factors like energy consumption, obstacle avoidance, and mission completion time. Additionally, PSO's iterative nature allows UAVs to continually refine their strategies based on real-time data, promoting robust performance in unpredictable scenarios. By integrating cognitive models with PSO, the decision-making process becomes more sophisticated, enabling UAVs to handle complex tasks with minimal human intervention. This approach not only improves the efficiency and reliability of UAV missions but also enhances their capability to operate in challenging environments, making it a preferred choice for advanced UAV flight algorithms.

In ARCog-NET, UAVs will be assumed to operate in an area $A \in \mathbb{R}^3$. The robots are represented like network nodes, being e the edge layer agents belong to the group $e = \{e_1, e_2, e_3, \dots, e_m\}$, and f is the group with the fog coordinator represented as $f = \{f_1, f_2, f_3, \dots, f_n\}$, where e_m and f_n are the maximum numbers of edge agents and fog coordinators, respectively. The assumptions for the framework modeling are that the transmission rate between components is constant, the processing power of each component is known and constant, and the latency and energy consumption depend on the amount of data processed and transmitted.

Time t_i is the initial moment of the operation in which the UAVs are deployed, the initial number of groups, agents per group, and coordinators assigned by the cloud supervisor HITL. Also, the initial trajectory points and flight trajectory deviation when an event happens can be set by the human supervisor at the beginning. It is important to notice that initially, fixed trajectories are considered for each agent and coordinator (calculated by the process loop or set by a human) in the ARCog-NET algorithm ($t = 0$), and flying trajectory modifications may occur at any time $t > 0$ throughout the operation. The number of objective-trajectory points at the beginning of the operation for the i-th drone assigned to a layer $L \in \{e, f\}$ is n^{L_i}, and the number of objective-trajectory points at any given time during the operation (t) is the variable $n_p^{L_i}$. This way, the number of objective points is:

$$n_p^{L_i} = \sum_{\lambda=1}^{M} \sum_{\gamma=1}^{N} \sum_{k=1}^{O} P_{L_i}^{\kappa}(x_\lambda, y_\gamma, z_k), \tag{1}$$

in which $t > 0$ is the moment where the trajectory is modified by any operation event, handled locally or by request/command in other layers, and $P_{L_i}^{\kappa}$ is the κ-th point of a UAV i in layer L executing the trajectory T_{L_i}. M, N and O are the maximum number of x, y and z coordinates of the points of this trajectory, therefore represents the total count of three-dimensional points in a finite set. The indices λ, γ, and k traverse the coordinates x, y, and z, respectively. $P_{L_i}^{\kappa}(x_\lambda, y_\gamma, z_k)$ indicates the presence of a point at the coordinate $x_\lambda, y_\gamma, z_k)$. Note that Equation 1 calculates the total number of three-dimensional points that the UAVs need to navigate to accomplish their tasks efficiently. This function is crucial because it defines how the UAVs' trajectories are determined and adapted in response to dynamic environmental factors, such as obstacles or changes in mission parameters. By optimizing these trajectories, the UAVs can ensure maximum coverage with minimal energy consumption and operational time, which is essential for the success of swarm-based autonomous systems. It is also important to know an incremental function denoted in Equation (2), which represents how much a point κ changes when it is modified by an event:

$$\Delta P^{\kappa}(t - t_i) = \begin{cases} \delta_p^{\kappa}, & \text{for } [||P_{L_i}^{\kappa}(t)_{\text{new}}|| - ||P_{L_i}^{\kappa}(t)_{\text{old}}||] \geq 0, \\ -\delta_p^{\kappa}, & \text{for } [||P_{L_i}^{\kappa}(t)_{\text{new}}|| - ||P_{L_i}^{\kappa}(t)_{\text{old}}||] < 0, \end{cases} \tag{2}$$

being $\delta_p^{\kappa} = |[||P_{L_i}^{\kappa}(t)_{\text{new}}||] - [||P_{L_i}^{\kappa}(t)_{\text{old}}||]|$ at time t, $P_{L_i}^{\kappa}(t)_{\text{new}}$ is the new κ trajectory T_{L_i} point and $P_{L_i}^{\kappa}(t)_{\text{new}}$ is the previous one. For a generic i-th flying robot assigned in a layer L (e or f) in the present framework, the position of a node's j-th "current objective" (doesn't matter the layer) at any given time t is $P_{\text{obj}_j}^{L_i}(t) \in \mathbb{R}^3$. Just for a mathematical generalization, it is considered that each robot flies toward its objective in a fixed height, which will be modified by random mission events by a Particle Swarm Optimization (PSO) algorithm. Actually, the objective's position depends on which task the swarm is applied, as it may be constant (trajectory flight for photogrammetry, for example) or suddenly modified by an operation event (the agent found an obstacle or received a trajectory modification command). The "objective motion" can be described as:

$$\begin{cases} \theta_{\mathrm{obj}_j}^{L_i}(t) = \arctan\left(\dfrac{P_{\mathrm{obj}_j}^{L_i}[y](t_i) - P_{\mathrm{obj}_j}^{L_i}[y](t)}{P_{\mathrm{obj}_j}^{L_i}[x](t_i) - P_{\mathrm{obj}_j}^{L_i}[x](t)}\right) \\ V_{\mathrm{obj}_j}^{L_i}(t) = \dfrac{||P_{\mathrm{obj}_j}^{L_i}(t) - P_{\mathrm{obj}_j}^{L_i}(t_i)||}{t - t_i} \\ P_{\mathrm{obj}_j}^{L_i}[x](t_{i+1}) = P_{\mathrm{obj}_j}^{L_i}[x](t) + V_{\mathrm{obj}_j}^{L_i}(t) \cdot \cos\theta_{\mathrm{obj}_j}^{L_i}(t) \cdot \Delta t \\ P_{\mathrm{obj}_j}^{L_i}[y](t_{i+1}) = P_{\mathrm{obj}_j}^{L_i}[y](t) + V_{\mathrm{obj}_j}^{L_i}(t) \cdot \sin\theta_{\mathrm{obj}_j}^{L_i}(t) \cdot \Delta t, \end{cases} \tag{3}$$

where $V_{\mathrm{obj}_j}^{L_i}$ denotes the "objective's speed" (or how fast the position of the j-th objective changed considering the UAV as an inertial referential), and $\theta_{\mathrm{obj}_j}^{L_i}$ is the "heading" of this objective (the direction in which this objective changed in relation to the UAV), both at any time where t. The robots may fly at variable altitudes in relation to the ground-relative level. Therefore, the position of the UAV at any time t and at any given layer (except cloud, which is not a UAV and is considered fixed) is denoted by $P_{L_i}(t) \in \mathbb{R}^3$. The robots fly between objectives through vectorial speed control considering constant height for mathematical simplification, which is modeled by the following equations:

$$\begin{cases} \theta_{L_i}(t) = \arctan\left(\dfrac{P_{L_i}[y](t_i) - P_{L_i}[y](t)}{P_{L_i}[x](t_i) - P_{L_i}[x](t)}\right) \\ V_{L_i}(t) = \dfrac{||P_{L_i}(t) - P_{L_i}(t_i)||}{t - t_i} \\ P_{L_i}[x](t_{i+1}) = P_{L_i}[x](t) + V_{L_i}(t) \cdot \cos\theta_{L_i}(t) \cdot \Delta t \\ P_{L_i}[y](t_{i+1}) = P_{L_i}[y](t) + V_{L_i}(t) \cdot \sin\theta_{L_i}(t) \cdot \Delta t, \end{cases} \tag{4}$$

in which $V_{L_i}(t)$ and $\theta_{L_i}(t)$ denote the instant velocity and the instant heading of any UAV in a given layer, respectively. It is assumed that $C_{L_i}(t)$ is the trajectory coverage at time t of navigated points P_{L_i} of a UAV in relation to its $P_{\mathrm{obj}_j}^{L_i}$ objective trajectory points. The coverage represents the trajectory traveled by a robot i between two consecutive moments t and $t + \tau$, where τ is a fixed time step. As T_{L_i} is the defined trajectory of waypoints performed (or to be performed) by an i-th UAV connected to ARCog-NET at layer L in an observed time t (being t_i the previous time), and $P_\kappa(t)$ is the κ-th point of this trajectory, the total coverage $C_{L_i}^{\mathrm{total}}$ of the trajectory can be described as:

$$\begin{cases} C_{L_i}^{\kappa}(t - t_i) = \Delta P_\kappa(t - t_i) \\ C_{L_i}^{\mathrm{total}} = \dfrac{1}{n_p^{L_i}} \sum_{k=0}^{n_p^{L_i}} C_{L_i}^{\kappa}(t - t_i), \end{cases} \tag{5}$$

where $n_p^{L_i}$ is the number of points observed in a time interval. This way, $C_{L_i}^{\mathrm{total}}$ will depend on how efficiently $n_p^{L_i}$ is defined, especially when navigation events happen (which could be a detected obstacle by an edge agent or a command to change trajectory by a superior layer). Thus, $C_{L_i}^{\mathrm{total}}$ can be understood as the reward for a navigation trajectory decision. When an event happens, by default, the method to calculate new trajectory points in the present work is the PSO algorithm [86].

ARCog-NET employs a cognitively improved PSO algorithm to calculate trajectory points of flight planning, improving classic PSO processing time and trajectory points precision. Basically, the trajectories executed by the UAVs are saved and used for building new trajectories using PSO algorithm. This way, improving the trajectory for new events becomes quicker if the calculation has to be done again. By integrating reinforcement learning into the PSO algorithm, we achieve a more adaptive and efficient trajectory path planning method for UAVs. The RL-enhanced PSO dynamically adjusts its parameters based on navigation feedback, leading to improved convergence rates and solution quality.

As robots must navigate through dynamic environments while avoiding collisions, by employing ARCog-NET PSO, these UAVs can quickly adapt their paths in response to changes in their surroundings, ensuring continuous operation without manual intervention. Similarly, in swarm operations, PSO enables each UAV to independently plan its path while maintaining formation integrity and avoiding both static and moving obstacles, or send a request to a superior layer for it to calculate a trajectory for it. This capability is critical for missions in complex terrains or urban environments, where the operational space is often cluttered and unpredictable. Also, ARCog-NET adds the functionality of comparing a new PSO trajectory with already calculated navigation decisions to verify if a previous decision is better than the current one calculated, repeating the previous decision of the historical database.

The ARCog-NET PSO is compared with classic PSO [87], A* [88], Dijkstra [89] and RRT [90] in convergence time, success rate, coverage, computational cost and scalability. Classic PSO, while effective, struggles with premature convergence and adaptability. ARCog-NET PSO improves this by using a cognitive navigation points base to enhance performance in dynamic environments, allowing it to be used in real-time applications. While A* excels in static pathfinding, it lacks adaptability for dynamic UAV operations. ARCog-NET PSO balances exploration and exploitation, offering superior performance and robustness for complex UAV swarm tasks. Dijkstra's algorithm is renowned for its ability to find the shortest path in graphs with non-negative edge weights, ensuring optimal solutions. By comparing ARCog-NET to Dijkstra, it is possible to highlight how closely ARCog-NET achieves optimal path solutions, particularly in static and well-defined environments. Finally, RRT (Rapidly-exploring Random Tree) is designed to efficiently explore large, high-dimensional spaces and is widely used in robotics for real-time path planning. By comparing ARCog-NET to

RRT, it's possible to illustrate ARCog-NET's capability to effectively explore and exploit search spaces in dynamic UAV environments.

Once the framework gives a connected robot the ability to access all knowledge base of other robots in its cluster (for edge points), another cluster (in case of fog), or the whole network (for cloud processing), improving the chance of finding the better solution, incremented with the possibility of consulting a human supervisor (i.e., through HITL) to guarantee a safe operation in cases that the network can't find a solution. The historic matrix of trajectory points coverage at a time t, $H_{L_i}(t)$, will present the coverage of the i-th UAV of layer L at a time t:

$$H_{L_i}(t) = \begin{bmatrix} C_{L_1}^{L'_1}(t) & \cdots & C_{L_n}^{L'_1}(t) \\ \vdots & \ddots & \vdots \\ C_{L_1}^{L'_m}(t) & \cdots & C_{L_n}^{L'_m}(t) \end{bmatrix}. \tag{6}$$

Therefore, when an operation event happens in time t, the decision to modify the trajectory of the i-th node belonging to L processed by the j-th node of L' is the one that presents the best-optimized coverage result. It is important to notice that a UAV will only have coverage registered for its processing and the processing performed in superior layers that it has communication (like a coordinator if it is an agent or the cloud if it is a coordinator), otherwise $C_{L_i}^{L'_j}(t) = 0$. The knowledge base of each UAV is then described by:

$$\kappa_{L_i} = \{H_{L_i}(t_0), H_{L_i}(t_1), \cdots, H_{L_i}(t_n)\}. \tag{7}$$

The knowledge base may start empty, with the robots making decisions locally at the first events while the knowledge base of each layer is improved, or consulting the HITL every time at the first events to start populating the base and reducing the need for consulting the human every time as the network becomes progressively smart. For each event that a decision should be made, there is a matrix for decision weights $W_{L_i}(t)$ associated to the matrix $H_{L_i}(t)$. This weight matrix can be described as:

$$W_{L_i}(t) = \begin{bmatrix} w_{L_1}^{L'_1}(t) & \cdots & w_{L_n}^{L'_1}(t) \\ \vdots & \ddots & \vdots \\ w_{L_1}^{L'_m}(t) & \cdots & w_{L_n}^{L'_m}(t) \end{bmatrix}, \tag{8}$$

in which $w_{L_i}^{L'_j}$ is the weight of the trajectory planning decision for the i-th robot at L commanded by node j at L'. This way, each element of a $H_{L_i}(t)$ matrix represents a coverage result of a trajectory calculated by an individual j in layer L' for the respective individual i in layer L (notice that the situation $L_i = L'_j$ may happen in case of a local coverage decision) and this element is related to the respective $W_{L_i}(t)$ weights matrix. The elements of the weight matrix represent the effectiveness of each coverage possibility obtained when calculating new trajectory coverage and are used as anchors for looking at the knowledge base. The weight history for this drone can be described as:

$$\omega_{L_i} = \{W_{L_i}(t_0), W_{L_i}(t_1), \cdots, W_{L_i}(t_n)\}. \tag{9}$$

Choosing a new trajectory when an event happens depends on how this decision is supposed to benefit one's mission, which is the optimal coverage process explained in Equation (5). If a processing layer produces the best coverage at a time t, it will have its $w_{L_i}^{L'_j}$ rewarded. Otherwise, it will be penalized, therefore modifying the vector of weights ω_{L_i}. This loop is a reinforcement learning process, which improves collective decision-making in the present system, giving better solutions privilege over ineffective solutions as the operation happens.

To sum up the reinforcement learning process, when an event happens to a vehicle i of layer L, this UAV will try to calculate a trajectory deviation to optimize current trajectory coverage. It will then look at the weights vector ω_{L_i} to see if a matrix of weights at any given time t, $W_{L_i}(t)$, produced an interesting decision weight $w_{L_i}^{L'_j}(t)$ that is higher than others. This weight will point to a previous time when an event also happened, and at that time, the event was processed by the node L'_j, and that should be accessed through the knowledge base vector κ_{L_i}, which will point to matrices of coverage history $H_{L_i}(t)$. Algorithm 1 represents the implemented path planning algorithm for dealing with navigation events.

If the coverage of that previous decision produced a better result than the result that the UAV just calculated, it will copy the result, improve its weight matrix, save it at the base, and update the weight vector, knowledge base, and historical matrix. Otherwise, the UAV can continue looking at the base, use the newly calculated result, or send a request to superior layers for a new decision. This process can scale up until the cloud, which will require the human supervisor for an intervention in the last instance. After the decision is taken and executed, the UAV will evaluate the real results, confirm the coverage, and keep the improved weight or penalize it, updating the last decision taken one last time. Figure 3 summarizes the reinforcement learning loop, and the diagram in Figure 4 represents the simplified cognitive algorithm of ARCog-NET.

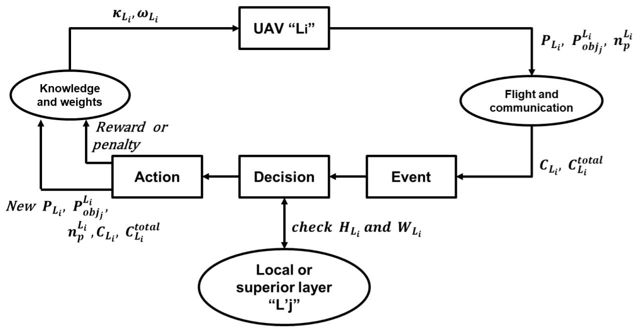

$\kappa_{L_i}, \omega_{L_i}$
UAV "Li"
$P_{L_i}, P_{obj_j}^{L_i}, n_p^{L_i}$
Knowledge and weights
Flight and communication
Reward or penalty
$C_{L_i}, C_{L_i}^{total}$
Action
Decision
Event
New $P_{L_i}, P_{obj_j}^{L_i}, n_p^{L_i}, C_{L_i}, C_{L_i}^{total}$
check H_{L_i} and W_{L_i}
Local or superior layer "L'j"

FIGURE 3. ARCog-NET reinforcement learning process of the cognitive decision-making loop.

Algorithm 1 Particle Swarm Optimization for ARCog-NET navigation event trajectory path planning

```
1:  Input: $T_{L_i}$: Initial trajectory positions of UAV $i$
2:  $V^{L_i}_{\mathrm{obj}_j}$: Initial velocities of trajectory points
3:  $\kappa_{L_i}$: Current knowledge base
4:  $\omega_{L_i}$: Current decision weights history
5:  $t_i$: Previous time
6:  $t$: Current time
7:  $C_{\min}$: Minimum coverage threshold
8:  $\max_{it}$: Maximum number of iterations
9:  $\min_{err}$: Minimum error threshold
10: while Error $>$ $\min_{err}$ and Iterations $\leq \max_{it}$ do
11:     for $PinT_{L_i}$ do
12:         $C_{L_i}(t - t_i) = \Delta P(t - t_i)$
13:         if $C_{L_i}(t - t_i) \geq C_{\min}$ then
14:             Trajectory keep ($P$)
15:         else
16:             Trajectory drop ($P$)
17:             New_Point = empty
18:             for a do in $\omega_{L_i}$:
19:                 for b do in $W_{L_i}$(rows):
20:                     for c do in $W_{L_i}$(columns):
21:                         Get BestFitting($w^{L'_i}_{L_i}$)
22:                     end for
23:                 end for
24:             end for
25:             $P \leftarrow \kappa_{L_i}[a]\{H_{L_i}[b, c]\}$
26:         end if
27:     end for
28:     Update Trajectory points:
29:     for $\kappa inrange(length(P))$ do
30:         Get best fitting $\kappa$-th $P$:
31:         $V^{L_i}{}_{\mathrm{obj}_{j_\kappa}}(t) = \frac{||P^{L_i}_{\mathrm{obj}_{j_\kappa}}[x](t) - P^{L_i}_{\mathrm{obj}_{j_{\kappa-1}}}(t)||}{t - t_i}$
32:         $\theta^{L_i}_{\mathrm{obj}_{j_\kappa}}(t) = \arctan\left(\frac{P^{L_i}_{\mathrm{obj}_{j_\kappa}}[y](t) - P^{L_i}_{\mathrm{obj}_{j_{\kappa-1}}}[y](t)}{P^{L_i}_{\mathrm{obj}_{j_\kappa}}[x](t) - P^{L_i}_{\mathrm{obj}_{j_{\kappa-1}}}[x](t)}\right)$
33:         $P^{L_i}_{\mathrm{obj}_{j_\kappa}}[x](t) = P^{L_i}_{\mathrm{obj}_{j_{\kappa-1}}}[x](t_i) + V^{L_i}_{\mathrm{obj}_{j_\kappa}}(t) \cdot \cos\theta^{L_i}_{\mathrm{obj}_{j_\kappa}}(t) \cdot \Delta t$
34:         $P^{L_i}_{\mathrm{obj}_{j_\kappa}}[y](t) = P^{L_i}_{\mathrm{obj}_{j_{\kappa-1}}}[y](t_i) + V^{L_i}_{\mathrm{obj}_{j_\kappa}}(t) \cdot \sin\theta^{L_i}_{\mathrm{obj}_{j_\kappa}}(t) \cdot \Delta t$
35:         if $|C_{L_{i\kappa}}(t)$ - $C_{L_{i\kappa-1}}(t)| \leq$ Error then
36:             Error $\leftarrow |C_{L_{i\kappa}}(t)$ - $C_{L_{i\kappa-1}}(t)|$
37:         end if
38:     end for
39:     Iterations $\leftarrow$ Iterations $+1$
40: end while
41: $t_i \leftarrow t$
42: Create new $H_{L_i}(t)$ matrix incorporating the new $C^{L'}_{L}(t)$
43: Incorporate new $H_{L_i}(t)$ to knowledge base $\kappa_{L_i}$
44: Create new $W_{L_i}(t)$ matrix incorporating modified $w^{L'}_{L}(t)$
45: Incorporate new $W_{L_i}(t)$ to weights history $\omega_{L_i}$
46: Output: $T_{L_i} = P_{\mathrm{optimized}}$: Optimized trajectory positions
47: $\kappa_{L_i}$: New knowledge base
48: $\omega_{L_i}$: New decision weights history
```

At the edge layer, the process starts by identifying an agent's particular tag number, followed by the retrieval of current UAV positions and objectives. These are used to calculate new objectives to optimize the trajectory coverage while accomplishing the mission. The proposed solution is then compared with an existing knowledge base. If an improvement is found, it's added to a solution vector and the knowledge base. Subsequently, an action is executed, and the effectiveness of the last action is evaluated, with adjustments made to the action weight based on the outcome. Suppose the agent doesn't find an improvement action in the knowledge base but finds a suitable action previously executed. In that case, it can repeat this solution, update its weight, and save it again as a new solution in the knowledge base. If the edge agent doesn't find a fitting solution for the operational event, it can request a new solution from the fog layer coordinator.

Simultaneously, the fog layer is engaged in a similar process. It checks for incoming requests from the edge and identifies relevant tags (its own, from its group of agents, or from other fog coordinators). Depending on whether it's processing an edge request or responding to a fog event, it calculates new objectives to optimize trajectory coverage. It then assesses the efficiency of the solution against its knowledge base. Efficient solutions are rewarded and updated in the solution vector and knowledge base. After sending decision data (in case of a cognitive processing request) or executing an action (in case of a self-fog-cognition usage), the fog layer evaluates the effectiveness of the last action and updates the weights and knowledge base of the last decision, concluding its cognition process.

Parallel to this, the cloud layer operates by recognizing requests or commands directed to it. When a layer L tag is detected, it gathers current UAV data points and objectives to compute new objectives that optimize that flight coverage. These objectives are weighed against the cloud's knowledge base. If a more effective solution is found, it's incorporated into the solution vector, updated in the knowledge base, and then executed. Otherwise, it should use a similar solution previously saved on the knowledge base (since the cloud has the most complete knowledge base of all layers). The process in the cloud layer wraps up with an evaluation of the new decision taken, updating weights, saving it to the knowledge base and other layers, and sending the processed action command to the requester.

Summing it up, the cognition of ARCog-NET can be modeled as a coverage optimization problem, in which the weights of each decision will help locate the best solution of an operational event in each node search or create a new solution if it doesn't find one. Each layer employs a systematic approach for continual learning and adaptation throughout the algorithm, incorporating feedback into their respective knowledge bases and adjusting their decision strategies accordingly. This mechanism allows the layers to operate independently and cohesively within the network's cognitive architecture.

It is important to point out that the treatment of the event is mainly directed to trajectory switching, and this modification may be caused by obstacles, communication failures, increasing latency rates, packet loss, and energy consumption, etc. In other words, UAVs can trigger new trajectory calculations or request networks to deal with this

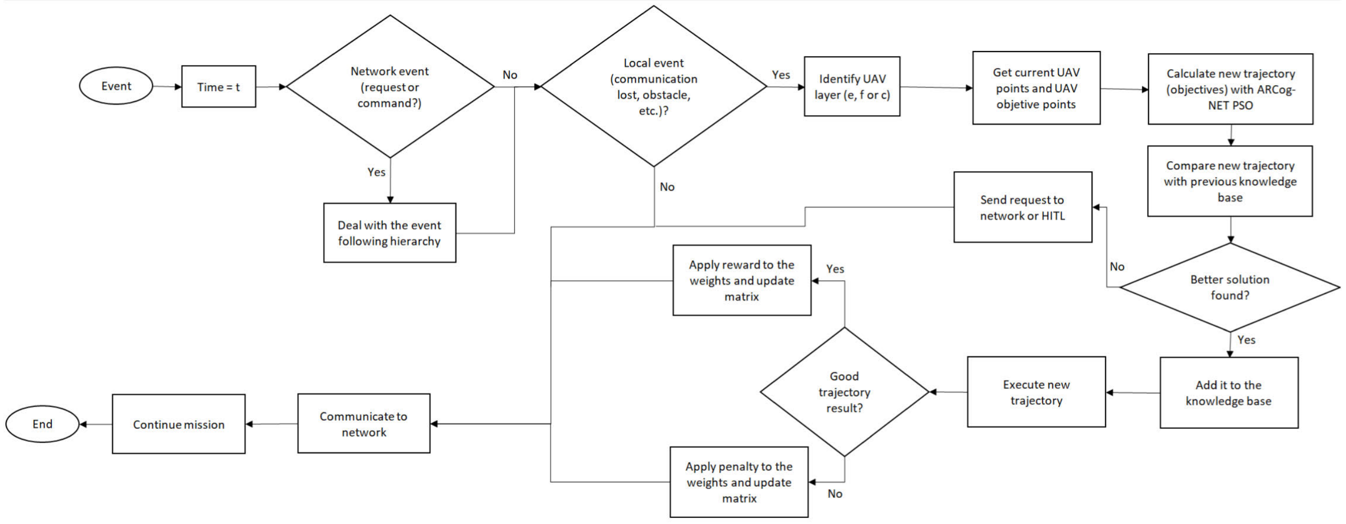

Event
Time = t
Network event (request or command?)
No
Yes
Deal with the event following hierarchy
Local event (communication lost, obstacle, etc.)?
Yes
No
Identify UAV layer (e, f or c)
Get current UAV points and UAV objetive points
Calculate new trajectory (objectives) with ARCog-NET PSO
Compare new trajectory with previous knowledge base
Better solution found?
No
Yes
Send request to network or HITL
Add it to the knowledge base
Execute new trajectory
Good trajectory result?
Yes
No
Apply reward to the weights and update matrix
Apply penalty to the weights and update matrix
Communicate to network
Continue mission
End

FIGURE 4. ARCog-NET flight cognition algorithm flowchart.

also based on communication or flight parameters. The network may also change the data routing pipeline, node's roles, number of UAVs in a group, or switch nodes between groups based on how the communication and flight behave, relying only on network management without changing the trajectories, but in this case, the presented models are not activated.

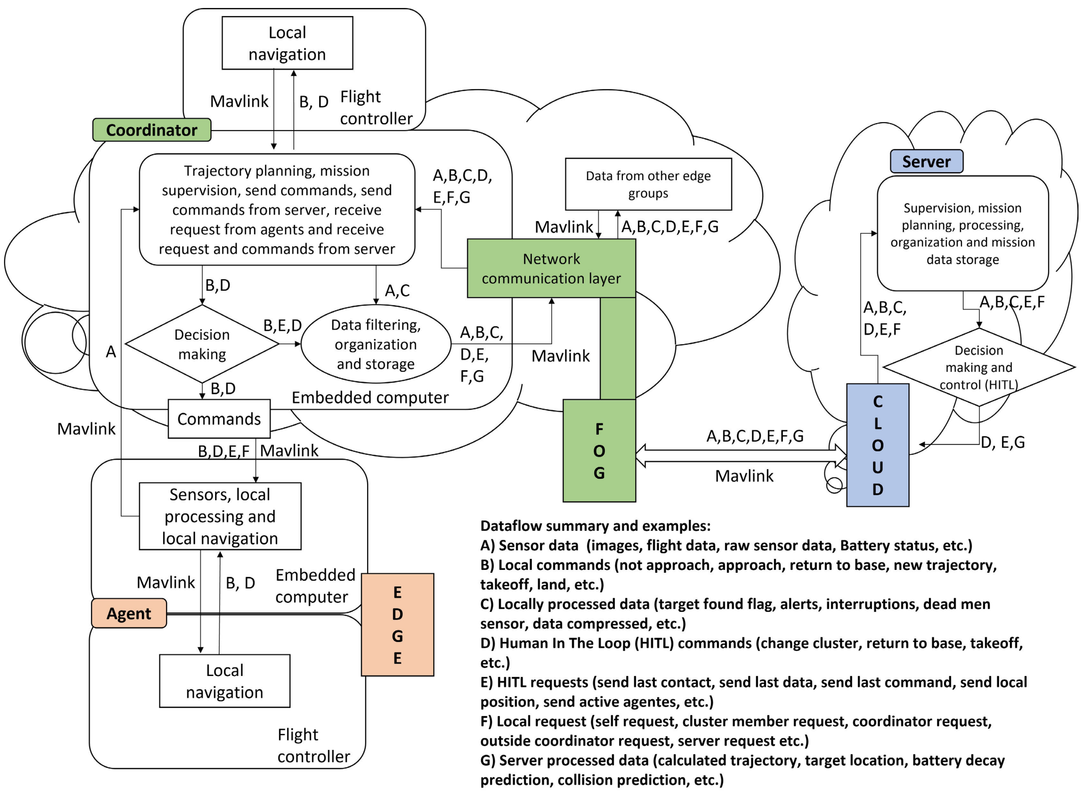
B. SIMULATION

Crafting a simulation architecture for ARCog-NET validation aims to enhance the development and testing of its usage in UAV swarm systems. Using ROS [69] and Gazebo [70] to provide a flexible framework for writing robot software and performing simulations has become a standard for researchers and developers working on autonomous systems. By incorporating Gazebo, a high-fidelity 3D robotics simulator, this architecture achieves realistic modeling of UAV dynamics and environmental interactions. When these capabilities are augmented with the MAVLink ROS (MAVROS) package [71], a ROS-based library that enables Micro Air Vehicle Link (MAVLink) protocol communication [72] between ROS and the ArduPilot/PX4 platforms [73], the simulation gains the ability to replicate real-world UAV behaviors and control mechanisms closely.

Based on the algorithms running at each ROS node on respective layers presented in previous topics, the data structures formatting between nodes, and the theoretical model of the network's cognition, the data flow within ARCog-NET architecture is one of the most important parameters to define the framework and practical operation of this tool. As previously known, the network is segmented into three kinds of primary nodes: the agent, coordinator, and server, each serving distinct functions and interconnected through data transmission channels. Although ARCog-NET can be used for developing general aerial robot swarm applications, this work presents the usage of the MAVLink as a standard transmission protocol and also for exchanging internal data between sensors and other modules. Due to its versatility for integration with real-life hardware [79], [80], lots of sensor plugin packages, and ease of modeling and integrating new sensors with MAVROS, this can be considered as a contribution of the architecture presented in this research. To understand ARCog-NET simulation methodology, it is crucial to know the framework's simplified dataflow represented in Figure 5.

Data flows seamlessly through this architecture, with the MAVLink protocol enabling communication across the network or between components. This ensures that sensor data, navigation commands, and other critical information are shared efficiently between the Agents, the Coordinators, and the Server. For instance, data generated by the Agents' sensors is sent to the Coordinator, who decides whether to act on the data locally or send it through the network to the Server for further analysis. The flowchart also outlines various data types and their flow paths. For example, sensor data (type A), local commands (type B), and HITL commands (type E) are channeled through different paths to ensure timely and appropriate responses within the network. The system's design facilitates robust and flexible UAV operations, enabling real-time responses to local events and integrating human oversight through the cloud for strategic decision-making.

The ARCog-NET is designed to have an FCU dedicated only to flight control, peripherals, and avionic data gathering, and all the data processing, decision-making, communication, and complementary operations that demand computational load may be executed on embedded boards like Raspberry Pi [91] or Galileo [92]. These offboard computers will command FCU through serial communication


Local navigation
Mavlink
B, D
Flight controller
Coordinator
Trajectory planning, mission supervision, send commands, send commands from server, receive request from agents and receive request and commands from server
A,B,C,D, E,F,G
Data from other edge groups
Mavlink
A,B,C,D,E,F,G
Network communication layer
B,D
A,C
Decision making
B,E,D
Data filtering, organization and storage
A,B,C, D,E, F,G
Mavlink
A
B,D
Embedded computer
Commands
Mavlink
B,D,E,F
Mavlink
F O G
A,B,C,D,E,F,G
Mavlink
Sensors, local processing and local navigation
Mavlink
B, D
Embedded computer
Agent
E D G E
Local navigation
Flight controller
Server
Supervision, mission planning, processing, organization and mission data storage
A,B,C, D,E,F
A,B,C,E,F
Decision making and control (HITL)
D, E,G
C L O U D

Dataflow summary and examples:
A) Sensor data (images, flight data, raw sensor data, Battery status, etc.)
B) Local commands (not approach, approach, return to base, new trajectory, takeoff, land, etc.)
C) Locally processed data (target found flag, alerts, interruptions, dead men sensor, data compressed, etc.)
D) Human In The Loop (HITL) commands (change cluster, return to base, takeoff, etc.)
E) HITL requests (send last contact, send last data, send last command, send local position, send active agentes, etc.)
F) Local request (self request, cluster member request, coordinator request, outside coordinator request, server request etc.)
G) Server processed data (calculated trajectory, target location, battery decay prediction, collision prediction, etc.)

FIGURE 5. ARCog-NET simplified dataflow representation.

while executing the algorithms presented in the present section. One of the advantages of this topology is the use of ROS packages that may run into the offboard controllers, with its simulation already being tested on Gazebo. The information may be exchanged between nodes and layers through free-to-air modulated signals using radio transmitter/receiver pairs. The cloud layer must have improved communication power, as it doesn't have hardware component constraints (since it is expected that it is a ground-based control unit) and may be farther from the cloud and edge UAV devices.

All development of ARCog-NET custom packages was made with Python 3.8.10 language, using mainly the libraries TensorFlow, Numpy, SciPy, sci-kit learn, OpenCV, raspy, and macros. The robot packages used ROS Noetic, and the simulations ran in the Gazebo 11 environment and with plugins. The main machine used for programming and simulation was a Lenovo IdeaPad Gaming 3i, with the following specs: 4,4GHz Intel Core i5 processor, 16 GB DDR4 3200MHz RAM, NVIDIA GeForce RTX 3050 4GB video board, 512GB SSD storage, and Ubuntu 20.04.6 LTS operating system. The simulated UAVs are all Yuneec Typhoon H480 hexacopters with PixHawk Px4 FCU, Raspberry 5 as an offboard embedded computer, and radio telemetry and communication modules, with the specs presented in Table 3. Table 4 shows the simplified hardware power consumption considered in the simulations. The performance of each mission was evaluated alongside the network parameters described in the next topics.

It is important to point out that ARCog-NET is built to use the communication protocol MAVLink as its base for exchanging messages between components and nodes, but the proposed topology of using embedded computer hardware coupled to the Flight Control Unit (FCU) allows compatibility with many existing network systems. This allows the proposed architecture to integrate into current network infrastructures without requiring significant modifications. The architecture can utilize existing communication channels like WiFi, LTE, 5G, and other wireless technologies to maintain connectivity and data flow between UAVs and other network components. The architecture supports communication with mobile devices through its cloud layer, which acts as a central hub for mission control and data analysis. Mobile devices, such as smartphones and tablets, can interface with the cloud layer to receive real-time updates, send commands, and monitor UAV operations (best option). Edge and fog layers may also have interfaces with mobile devices for exchanging real-time data directly from the respective nodes (agents and coordinators), but this can cause a latency increase in these UAVs due to processing load and also more power consumption.

In the simulation, Agents (edge) represent individual UAVs equipped with sensors and embedded computers. They handle local processing tasks like navigation and sensor data analysis. The processed data can include images, flight data, and sensor outputs, which are then sent to the Coordinator (fog) for further decision-making or directly to the Server for storage and higher-level analysis. At the core of this architecture, the Coordinator is responsible for trajectory planning and mission supervision. It sends and receives commands to and from the Server (cloud) while also handling requests from Agents. This node processes data such as sensor inputs and navigation commands, making decisions that are then communicated to the Agents and the Server for cohesive mission execution. The Server, operating within the cloud layer, oversees the entire mission, providing additional processing power for data organization and long-term storage. It also plays a crucial role in decision-making, particularly with HITL commands, where a human operator can intervene and control the swarm's actions, ensuring mission flexibility and adaptability to changing conditions.

The main idea adopted for developing the UAVs' simulation was to launch separated agents with the PX4 FCU and develop custom nodes representing offboard controllers. These offboard nodes will communicate to their respective FCU through MAVLink protocol (in this case, MAVROS) to receive data from sensors, alarms, status, position, velocity, and any other variable that is important for the

TABLE 3. Communication and processing power specifications.

Device	Pixhawk PX4 FCU	Raspberry Pi 5	Radio Telemetry
Processor	STM32F427, Cortex M4, 168 MHz	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8), 1.5GHz	N/A
Memory	2 MB Flash, 256 KB RAM	4 GB, 8 GB LPDDR4-3200 SDRAM	N/A
Comm. Inter-faces	UART, SPI, I2C, CAN, USB	Gigabit Ethernet, WiFi 5 (802.11ac), Bluetooth 5.0, BLE	LTE, HSPA+
Data Rates	N/A	Ethernet: up to 1 Gbps, WiFi: up to 866.7 Mbps	LTE: up to 150 Mbps (Download), up to 50 Mbps (Upload)
Frequency Bands	N/A	2.4 GHz, 5.0 GHz bands	Multiple bands depending on carrier and region
Additional Features	Advanced sensor integration for navigation, Multiple safety features including triple redundant IMU	Extensive GPIO and peripheral support, 4K video support	High mobility support, designed for IoT applications

TABLE 4. Power consumption of different hardware components.

Hardware	Power
Embedded Computer	27 W
UAV	120 Wh
Radios	1 W
Mobile Communication Modules and Antenna	100 W
Cloud	65 W

operational controller of each robot. In real life, this offboard controller may be a companion computer like a Raspberry Pi attached to the UAV, and the communication between the UAV and the companion computer may happen through serial communication via USB or the MAVLink telemetry radio (one attached to the FCU and one attached to the embedded computer). In the offboard controller, packages for controlling its respective UAV may be developed, but also for communicating with the simulated network and for using a different sensor that can't be embedded in the UAV's FCU. For example, this sensor could be an RGB camera, which in simulation is a Gazebo external plugin streaming the image "seen" by each UAV through a pipeline using the Python GStreamer library and OpenCV, transformed to a camera node that communicates to each offboard node. The active ROS nodes diagram for a 2 UAV group (one edge agent and one fog coordinator) developed under ARCog-NET is presented in Figure 6.

Besides the sensor data, an example of data processing is the image compression autoencoder model used for image compression and decompression in simulation. It is also integrated into the offboard nodes, taking the camera's image from the camera nodes and transmitting it to the network. In real life, data communication may be addressed with radio transceivers, WiFi, ZigBee, LoRa, or other wireless communication technology associated with the companion computer. The simulated UAV was a Typhoon H480 model, and the simulation was performed with groups of 1 to 10 vehicles. The next step in creating the simulation in ROS and Gazebo to test the ARCog-NET packages and behavior was to create generic missions to receive applications that could be assigned to a UAV group. A mission was designed, for that matter, being this scenario a representation of a vertical wind turbine field, where the UAVs should take off from a land base and fly through wind turbines for predictive maintenance surveillance using coverage path planning algorithms.

The simulation environment developed for this first application and the UAVs experimented are presented in Figure 7, and here, the major concern is autonomous navigation and interference between communications. The goal of this simulation is to check the capability of the ARCog-NET framework to plan and execute the task of flight between a known mesh of relatively close obstacles while avoiding collisions between robots and these objects, reducing communication interference, latency, and package loss between layers and allow a good level of data throughput

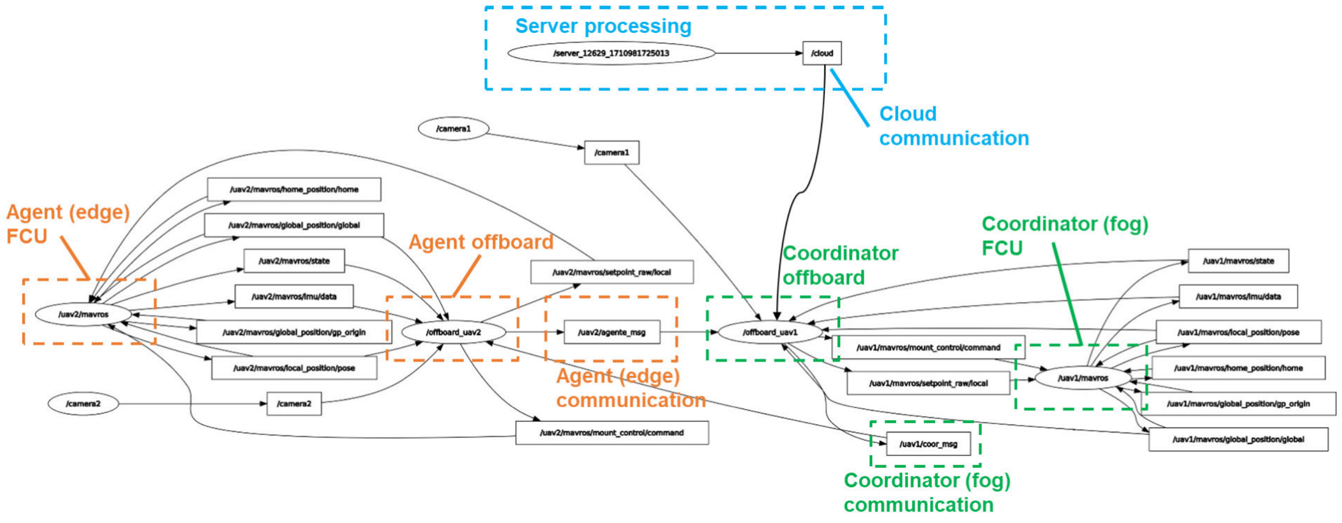

Server processing
/server_12629_1710981725013
/cloud
Cloud communication
/camera1
/camera1
Agent (edge) FCU
/uav2/mavros/home_position/home
/uav2/mavros/global_position/global
/uav2/mavros/state
/uav2/mavros/imu/data
/uav2/mavros
/uav2/mavros/global_position/gp_origin
/uav2/mavros/local_position/pose
/camera2
/camera2
Agent offboard
/offboard_uav2
/uav2/mavros/setpoint_raw/local
/uav2/agente_msg
Agent (edge) communication
/uav2/mavros/mount_control/command
Coordinator offboard
/offboard_uav1
Coordinator (fog) FCU
/uav1/mavros/state
/uav1/mavros/imu/data
/uav1/mavros/local_position/pose
/uav1/mavros/mount_control/command
/uav1/mavros
/uav1/mavros/home_position/home
/uav1/mavros/setpoint_raw/local
/uav1/mavros/global_position/gp_origin
/uav1/coor_msg
/uav1/mavros/global_position/global
Coordinator (fog) communication

FIGURE 6. Active ROS nodes and topics for 2 UAV and a serve in simulation.

because the robots are being used to inspect the wind turbines structures for predictive maintenance.

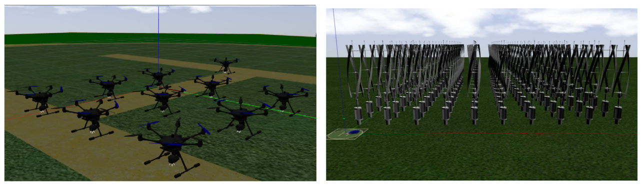


FIGURE 7. Swarm of 10 Typhoon H480 and vertical wind farm in simulation.

Theoretically, the fog coordinator can manage a group of 10 Edge agents in ARCog-NET, depending on its computational capacity and the network's requirements. The nodes handle commands, requests, and data through queue management, as presented in 2. Fairness is maintained using fair scheduling algorithms, load balancing, priority adjustment, and real-time feedback mechanisms. This ensures that all Edge agents are treated equitably and the overall system operates efficiently. Mathematically, ARCog-NET has no fog coordinator limitation, depending on the processing hardware, transmission and protocol limitations.

The simulation is executed with one to ten vehicles connected in ARCog-NET, 10 times for each connected robot count, being 100 simulations in total. The server (cloud layer) is initially configured to request the structure to use some of those robots as fog nodes (which can vary from 1 to 10 robots), which are also edge group coordinators. The number of the other edge agents may vary throughout the mission, so a fog coordinator may have up to 9 agents connected to it. While flying to seek wind turbine pathology, the UAVs are subjected to a collision between each other and the structures, and the ARCog-NET capabilities are used to calculate new trajectories and deviations. The robots also may

Algorithm 2 ARCog-NET Client Setup and Message Handling

```
1:  Define message_id, message_type, sender_id,
      receiver_id, date, message_content
2:  Initialize ARCog-NET client
3:  **Load** queue
4:  Connect to receiver_id
5:  **procedure** OnMessage(receiver_id, sender_id, message,
      message_type):
6:      **if** message_type = command **then**
7:          HandleTaskRequest(message.payload)
8:      **else if** message_type = request **then**
9:          ProcessStatusUpdate(message.payload)
10:     **else if** message_type = data **then**
11:         ProcessData(message.payload)
12:     **end if**
13:     **Update** queue
14: **end procedure**
15: Set message handling function: OnMessage
16: Subscribe to sender_id, message_type, message_content
17: Start ARCog-NET client loop
```

pre-process sent images (compression) and request superior layers to process decisions or for HITL decisions.

It is important to note that the initial delay in starting a mission with the UAV swarm typically arises from the need to initialize the UAVs (i.e., establish communication links, and initialize onboard sensors and systems). Thus, this delay in starting the mission may vary depending on the mission's complexity and the number of UAVs involved. Each UAV must establish a stable communication link, exchange initial data, and confirm its position and status within the swarm. The increased number of UAVs also places a higher computational load on the fog and cloud layers for processing, potentially contributing to further delays. However, ARCog-NET is designed to mitigate this issue by efficiently distributing computational tasks across the network layers, ensuring that mission initiation remains within acceptable timeframes even as the swarm size grows.

IV. SIMULATION RESULTS

To exemplify trajectories planned and executed by a group of 10 UAVs in a simulation executed with this UAV counting, a video was captured and presented in [93]. Figure 8 shows these executed trajectories, performed at the same time, in a cartesian plan graphic.

It is possible to notice how the initially calculated trajectories graph, which was planned using the PSO algorithm presented, shows that on certain occasions, some of the UAVs would collide with an obstacle (a vertical wind turbine, represented in the graph as black points). Alternatively, on the navigated trajectories graph, it is possible to notice the evasive actions performed by the UAVs while accomplishing the mission, also calculated with the PSO decision-making loop presented in the Methodology. The obstacle avoidance is essentially the decision to fly around the obstacle, bypassing it "above" or "below" on the *xy*-plan. As exposed in Section III, this obstacle avoidance event decision can be made in each layer by a robot or in its superior layer by request.

Note that, unlike typical drones that rely on reactive obstacle avoidance, ARCog-NET encompasses perception, planning, decision-making, and adaptive learning. This integration allows UAVs to not only react to obstacles but also predict and plan their movements intelligently based on past experiences and real-time data. Besides, its reinforcement learning capability allows UAVs to improve their decision-making process continually. This means that UAVs learn from each mission, enhancing their obstacle-avoidance strategies over time.

The performance of the ARCog-NET algorithm was also compared with other classic path-planning strategies, as mentioned before. Table 5 shows the results of a mathematical simulation of the UAVs using different types of implementation. Overall, ARCog-NET PSO excels in balancing convergence speed, success rate, computational cost, and scalability. It improves upon classic PSO and outperforms Dijkstra's in most metrics while remaining competitive with A* and RRT, making it a robust choice for UAV swarm path planning in dynamic environments.

Figure 9 shows the number of decisions made at each layer according to the number of UAVs on each simulation.

With a single UAV, the structure becomes only edge-cloud or only fog-cloud (depending on the cloud's configuration, but in practice, these two topologies have the same capabilities). With more than 1 robot, the structure will necessarily have an edge node and a fog node alongside the cloud layer. That being said, the graph suggests variability in the number of decisions processed at different layers, likely showing how

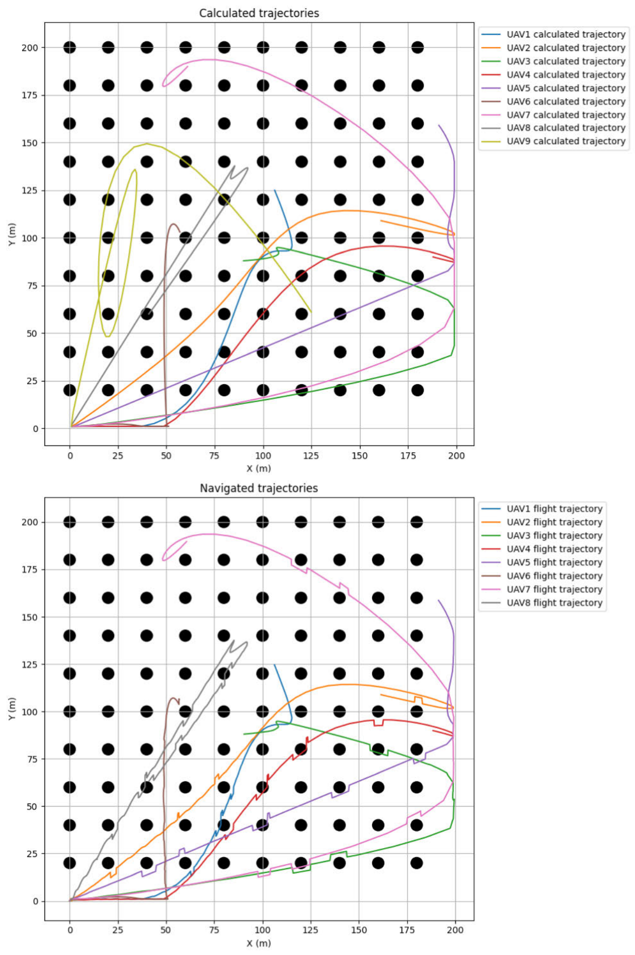


FIGURE 8. UAVs flying between wind turbines.

increasing the number of UAVs impacts the decision load at each layer. Most simple decisions are expected to be made in the edge layer without sending processing requests to the layers above. At 2 UAVs, it is possible to see an interesting behavior of ARCog-NET, as one robot is an edge node and the other is a fog (and they can change roles as the mission unfolds), the median number of processed decisions are very close in these two layers for this number of UAVs on the mission, with the edge agent having more processing than the fog coordinator.

Overall, most of the decisions are taken with cloud processing, which is an expected result as this layer has more computer power and access to all knowledge bases of the entire structure, making this layer more efficient for the cognitive reinforcement learning decision-making process. The layer where the decision-making loop is processed also affects latency, alongside a number of connections (capacity), distance, interference, and other parameters presented in Section III. Although the number of decisions processed in cloud and fog increases with more UAVs in simulation, the average latency decreases each time and becomes more stable. This represents how the proposed architecture of

TABLE 5. Numerical comparison of path planning algorithms.

Metric	ARCog-NET PSO	PSO	A*	Dijkstra's	RRT
Average Convergence Time (s)	2.5	5.2	1.8	10.3	3.6
Success Rate (%)	89	85	90	80	88
Path Optimality (% of optimal)	90	90	99	99	85
Computation Cost (CPU ms)	120	200	50	400	100
Scalability (nodes)	1000	800	500	300	900

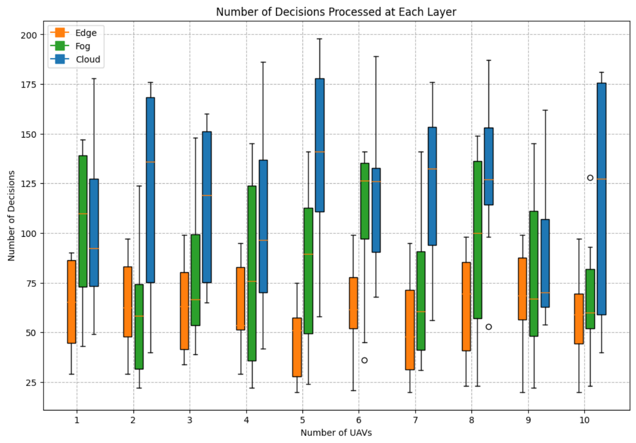


FIGURE 9. Number of decisions processed on each ARCog-NET layer in the wind field simulation.

distributed data processing and decision-making algorithms allows more efficient missions when analyzing the big picture. The whole simulation average latency with respect to the number of UAVs is presented in Figure 10.

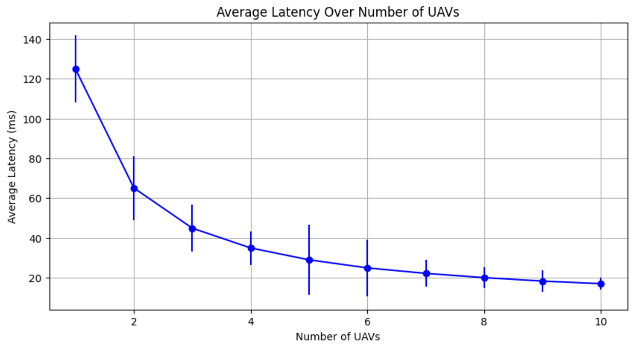


FIGURE 10. Average ARCog-NET latency by number of UAVs in the wind field simulation.

This graph shows that ARCog-NET also has a latency saturation as the number of UAVs in the network increases. The stability of the architecture in terms of latency, represented as the error bar at each point, also increases as more robots are added as network nodes. This is also a logical result, considering that more robots represent more connections, more processing power, less distance between each node, and also a larger knowledge base of decisions. With the ARCog-NET asset to automatically and dynamically set up its structure (robots designed as edge and fog nodes) and the connections between each layer during the mission, this result represents an excellent theoretical way to show that ARCog-NET is suitable for known environments outdoor missions with a mesh of objects involved.

With more UAVs connected and executing a mission, more data will flow through network nodes, and more data processing and decision-making available computing power, data routing, and fewer buffer queues. This decreases the latency, as presented before, and gives ARCog-NET more capacity to deal with interference and momentary communication loss. However, the throughput graph of Figure 11 also shows a throughput saturation of the proposed architecture after the number of 8 UAVs are added to the architecture in this simulation scenario. This represents that the framework probably reached its capability of processing, dealing with data, routing network communications, and accomplishing the given task while using the simulated hardware.

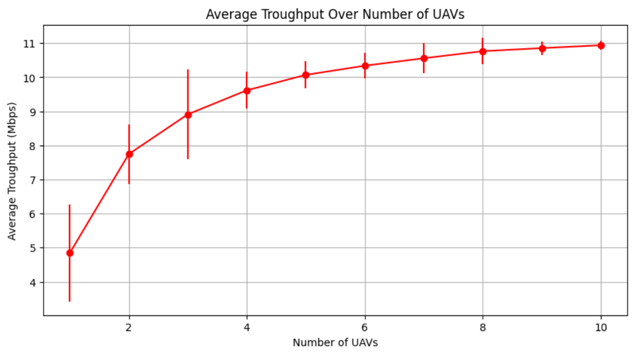


FIGURE 11. Proposed architecture's average throughput over the number of UAVs in operation in the wind field simulation.

The next step in evaluating and confirming the ARCog-NET's viability is to discuss the quality of service (QoS). Latency and throughput in this first simulation have already shown that the proposed architecture has good capabilities of controlling coordinated group flight behavior while dealing with communication, but it also lacks optimization for working properly with the full hardware and communication technologies that each UAV is equipped with in simulation. While the edge and fog UAVs or cloud server are busy dealing with incoming data, processing data from buffer queue, or even packing information for streaming, it may occur data packet loss, which represents a good indicator of QoS.

For calculating packet loss, the number of packets transmitted by each UAV is monitored and compared to the number of packets successfully received by the intended recipient. Packet loss is determined by the difference between these two values. This calculation considers all types of packets, including control messages, data packets, and any other communication relevant to the UAV's operations. The size of packets in the ARCog-NET system depends on the type of data being transmitted. Typically, the packets can range from small control messages of a few bytes to larger data packets that might include image or video data. For control messages and telemetry data, packet sizes are relatively small, usually around 20 to 100 bytes. For image data, especially high-resolution images or video streams, packet sizes can be larger, but not exceeding 256 bytes, depending on the compression and the resolution of the data.

The critical situation is when just one UAV is used with the proposed framework, where packet loss seems to reach its peak. This result is caused by the lack of distributed processing, node communication, and small knowledge base formation, as just one network node (or robot) operates with the cloud layer. Figure 12 shows the average packet loss for each number of UAVs connected at ARCog-NET in the wind field simulation.

Besides, the percentage of packets lost is very low. This result confirms that, in fact, for this scenario, there is a saturation in ARCog-NET capabilities that do not scale up, even if more robots are added to the network. This also shows that the proposed architecture performs relatively well for a single-robot ground control base but is very inefficient compared to the results the user may get when more UAVs are added. In practical terms, this architecture may represent a more complex way to control only one robot in a mission. The package loss results are also in line with latency and throughput readings, presenting both the performance and stability of ARCog-NET.

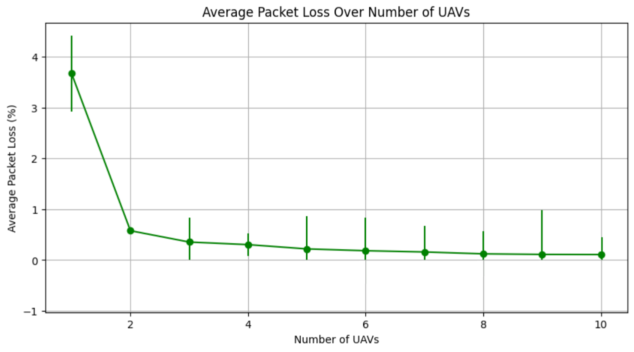


FIGURE 12. ARCog-NET framework average packet loss over a number of UAVs in operation in the wind field simulation.

One important aspect of analyzing is data compression when discussing the network performance associated with the robot's performance to accomplish the objective. For this particular mission, each UAV image is important as the user

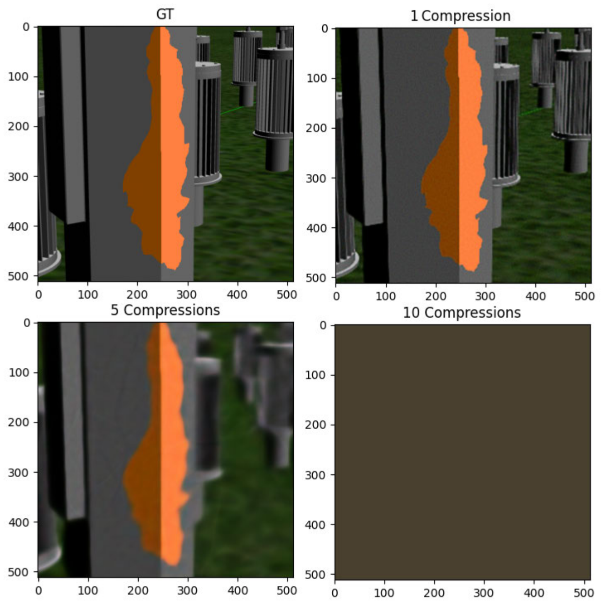


FIGURE 13. Compression example of a picture of corrosion found at a wind turbine shaft.

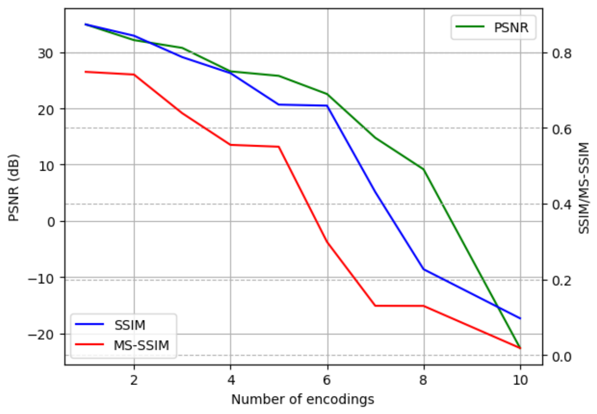


FIGURE 14. PSNR, SSIM, and MS-SSIM variation for interlinked compression in wind field simulation.

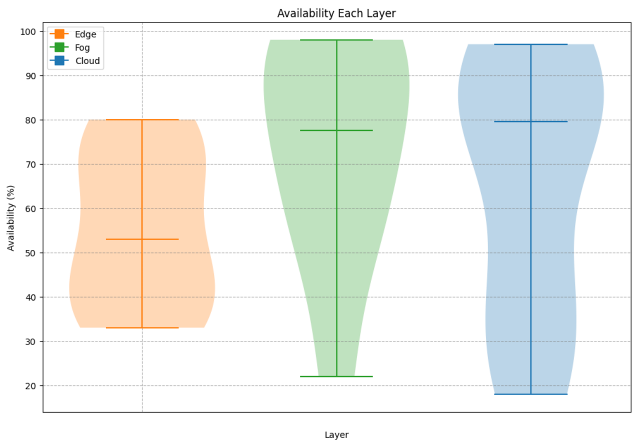


FIGURE 15. Availability of each layer in the wind field simulation for ARCog-NET.

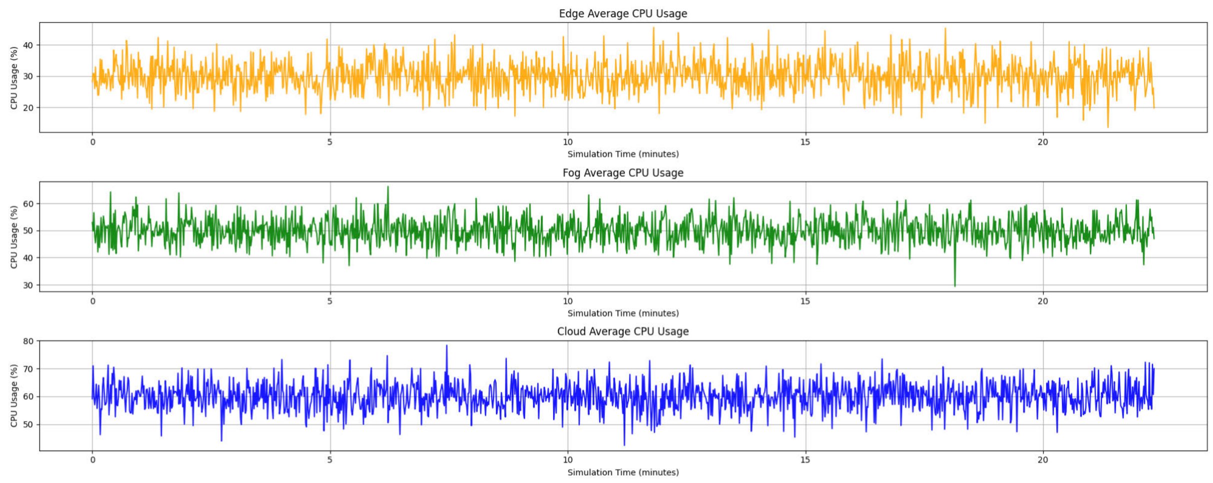


FIGURE 16. **CPU usage of each layer at each time instant in the wind field simulation.**

(supervisor) HITL may identify corrosion or other kind of damage at the wind turbines being inspected. As presented in Section III, there is a built-in data compressor for images in ARCog-NET, which is an autoencoder that allows successive compression and decompression stages in a way that the data can be streamed without any information loss (raw data) or greater losses that the end-user may allow (users set which compression level is more interesting to themselves, and this asset is reconfigurable during the mission). The compression will also affect latency, throughput, and packet loss because the more the data is compressed, the fewer transmission parameters are demanded. Still, more processing loads are required (for data compression and decompression). The details of the implemented autoencoder, training, and evaluation results can be found at [23], developed by the authors of the present study. A compression result example is presented in Figure 13, which shows an image captured from one of the robots during the wind field simulation detailing corrosion found at one of the wind turbines' rotary shafts and how this image is affected by successive compression with the trained autoencoder.

It is possible to notice that it's still possible to identify damage at the structure even with five successive compressions. However, at 10 compression levels, the picture becomes unrecognizable, affecting the mission objective. Also, this example shows big and easy-to-identify damage, so smaller ones may also be affected by less compression. The model compressing and decompressing computation is fast, so the byte volume transmission reduction (with the cost of losing data) can be achieved by compressing the image several times before streaming it. The number of compressions for each UAV transmitting image could be defined in the offboard controller nodes, which is the group's collective decision on which agent should detail more or less the data they're sending. In this simulation case, the number of compressions is a decision made by the cloud supervisor HITL. In order to validate the autoencoder's performance, images from the UAVs were used during the simulation, being compressed from 1 to 10 times, and the Peak signal-to-noise ratio (PSNR), Structural Similarity Index (SSIM), and Multi-Scale SSIM (MS-SSIM) were computed. Figure 14 presents the variation results for each indicator.

The similarity of the compressed image decreases with successive compression as expected, and the signal/noise relation also decays fast. After the seventh compression, the image is practically unrecognizable. Even though image data compression may improve network performance, it should not be used indiscriminately as it may also hinder the mission's goals. Figure 15 is a violin plot of the average availability of each layer of ARCog-NET observed during the wind field simulation.

Except for the edge layer, the graph shows that the layers have good availability as the wider areas of the violin are above 60% of availability, and fog and cloud have 79.16% and 78.56% of average availability, respectively. The edge layer represents a concern, as the areas are equally distributed over higher and lower availability, and the average availability of this layer is 53.11%, which is a relatively low result. In fact, the edge layer seems to be more affected by the number of UAVs than the fog, and the processing algorithm in this layer can be improved. This result may cause scalability concerns for increasing robots as edge nodes, which will be addressed in future research reviews. However, ARCog-NET can be considered a feasible option for control and communication for UAV swarms operating in known outdoor environments. The other factor that has a combined influence on the network and the robot's performance is power consumption. Figures 16 and 17 present the Central Process Unit (CPU) usage percentage at each time instant and the power consumption (considering UAV, transmission, and processing energy) for the simulations using 10 UAVs, which are the worst case scenario for evaluating this parameter.

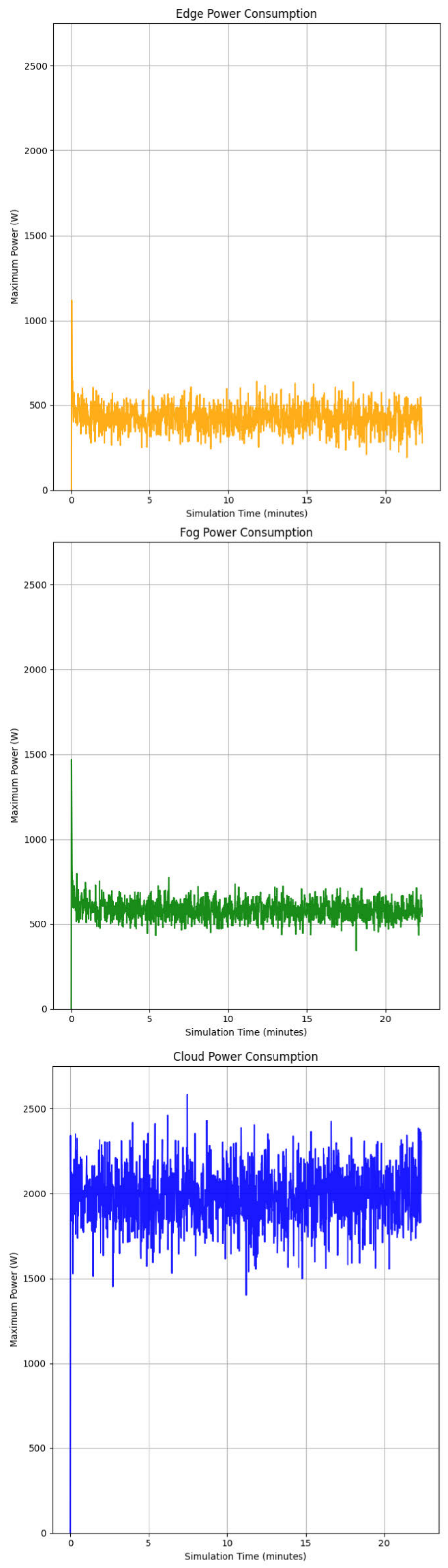


FIGURE 17. Power consumption of each layer at each time instant in the wind field simulation for the proposed framework.

The CPU usage for the edge layer is 30.18%, for the fog layer is 50.03%, and for the cloud 59.91%. As edge and fog have similar computational capabilities, the big difference in CPU usage can be explained by the fact that edge agents have less data processing to do, as these robots are more dedicated to collecting and preprocessing data and fly safely through the environment, as fog nodes have edge group coordinators roles, data processing, filtering, and midway storage while also flying safely. By looking only at CPU usage, the cloud layer seems similar to fog, but this processing is not at the same scale. Cloud computing capabilities are much larger than those of the embedded systems of UAVs, and in fact, this can be seen in the power consumption graph. Edge and fog, even with the robot nodes, have lower power consumption levels than the cloud, which has only computers and radio communications. At the power consumption graphs, it's also possible to see the initial peak for edge and fog layers, representing the takeoff energy drained by the group. Of course, the power drained from each UAV and by the cloud server will not have the same saturation observed in other parameters for this first simulation, as more robots in the group represent more energy demand added to the network.

Regarding scalability, the simulations with varying numbers of UAVs demonstrated that ARCog-NET could efficiently manage up to 10 UAVs. Initially, with fewer UAVs, most decisions are made at the edge layer. As more UAVs are added, the fog and cloud layers take on a larger share of the decision-making process, effectively managing the computational load and preventing any single layer from becoming a bottleneck. However, further testing with more scenarios and a larger number of UAVs is necessary. In the tests, each UAV performed its tasks effectively while maintaining communication and coordination with other nodes. The results showed that with an increased number of UAVs, the architecture could handle more complex scenarios with higher data throughput and improved operational effectiveness. In this sense, the results also show once again that ARCog-NET is a suitable architecture for robot swarm development in outdoor known environments applications, but it still has optimizations to explore.

V. CONCLUSION AND FUTURE WORK

The proposed framework in this paper, entitled ARCog-NET, not only facilitates improved collaboration among UAV swarms by incorporating advanced data management and cognitive decision-making processes but also integrates with existing network structures, ensuring efficient operation and task execution by the UAVs. Notably, one of the standout features of ARCog-NET compared to other cognitive architectures of the literature is its use of the EFC model for distributed processing. This model enables efficient distribution of computational tasks, where real-time data collection and initial decision-making occur at the edge level (UAV level), intermediate processing happens at the fog level (group of UAVs), and complex analysis and long-term data storage are managed at the cloud level. This hierarchical structure significantly reduces latency, allowing UAVs to react more swiftly to obstacles. Additionally, it enables the system to scale with the number of UAVs in the swarm, ensuring optimal performance in complex environments.

The implementation of ARCog-NET, in conjunction with ROS and tested in the Gazebo simulation environment, underscores the work's contribution to the practical application of cognitive architectures in UAV swarms. The detailed testing and evaluation revealed not only the functionality and performance improvements by ARCog-NET but also the potential for future advancements in UAV swarm operations, particularly in terms of communication efficiency, operational reliability, and mission effectiveness.

At this point, it is a fact that the proposed architecture seems to be suitable as a swarm robot controller architecture, but it works below the full simulated hardware capabilities. To improve the proposed architecture performance, more simulation scenarios and applications may be executed and analyzed. Also, more communication optimizations can be studied and implemented, such as dynamic data routing and smart group clustering (currently, it uses only proximity between robots). Another factor that can make results more reliable is using network communication simulators such as Omnet to evaluate communication parameters.

As ROS packages are ready to deploy in physical hardware, this suggests that the proposed architecture is also ready for field tests, which would be an important step in confirming ARCog-NET as a solution for developing UAV swarm applications, which is its main objective. Finally, it is possible - under some adaptations - to use ARCog-NET in different and heterogeneous robot groups, as the framework was developed mainly based on PixHawk Px4 hardware, and this FCU can be used in terrestrial, aquatic, and underwater robots as well, not to mention different aerial robots typologies.

In terms of evaluation, this research work opens up several future possibilities. For instance, as more end devices can be added to the system, the latency can increase due to the additional processing load on the edge and fog layers, being a critical issue for real-time operations where prompt decision-making is crucial. A future implementation will involve the use of a dynamic load-balancing mechanism across the edge and fog layers in order to allocate processing tasks more intelligently by prioritizing critical real-time data while offloading less time-sensitive tasks to cloud layers or secondary nodes.

ACKNOWLEDGMENT

The authors would like to thank the following Brazilian Federal Agencies, CNPq, FAPERJ, and CEFET-RJ, for supporting this research work.

REFERENCES

[1] J. Parikh and A. Basu, *Unmanned Aerial Vehicles: State-of-the-Art, Challenges and Future Scope*. Hoboken, NJ, USA: Wiley, 2021, ch. 2, pp. 29–42.

[2] G. S. Ramos, M. F. Pinto, F. O. Coelho, L. M. Honório, and D. B. Haddad, "Hybrid methodology based on computational vision and sensor fusion for assisting autonomous UAV on offshore messenger cable transfer operation," *Robotica*, vol. 40, no. 8, pp. 2786–2814, Aug. 2022.

[3] M. F. Pinto, A. G. Melo, Andre. L. M. Marcato, and C. Urdiales, "Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle," in *Proc. IEEE 26th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2017, pp. 1324–1329.

[4] G. S. Ramos, D. B. Haddad, A. L. Barros, L. de Melo Honorio, and M. F. Pinto, "EKF-based vision-assisted target tracking and approaching for autonomous UAV in offshore mooring tasks," *IEEE J. Miniaturization Air Space Syst.*, vol. 3, no. 2, pp. 53–66, Jun. 2022.

[5] C. D. Rodin, L. N. de Lima, F. A. de Alcantara Andrade, D. B. Haddad, T. A. Johansen, and R. Storvold, "Object classification in thermal images using convolutional neural networks for search and rescue missions with unmanned aerial systems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[6] G. S. Berger, M. Teixeira, A. Cantieri, J. Lima, A. I. Pereira, A. Valente, G. G. R. D. Castro, and M. F. Pinto, "Cooperative heterogeneous robots for autonomous insects trap monitoring system in a precision agriculture scenario," *Agriculture*, vol. 13, no. 2, p. 239, Jan. 2023.

[7] L. D. P. Pugliese, F. Guerriero, and G. Macrina, "Using drones for parcels delivery process," *Proc. Manuf.*, vol. 42, pp. 488–497, Jan. 2020.

[8] D. R. Green, J. J. Hagon, C. Gomez, and B. J. Gregory, "Using low-cost UAVs for environmental monitoring, mapping, and modelling: Examples from the coastal zone," in *Coastal Management*, R. Krishnamurthy, M. Jonathan, S. Srinivasalu, and B. Glaeser, Eds., New York, NY, USA: Academic, 2019, pp. 465–501.

[9] S. Manfreda, P. Dvorak, J. Mullerova, S. Herban, P. Vuono, J. A. Justel, and M. Perks, "Assessing the accuracy of digital surface models derived from optical imagery acquired with unmanned aerial systems," *Drones*, vol. 3, no. 1, p. 15, Jan. 2019.

[10] A. Tahir, J. Boling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles—A survey," *J. Ind. Inf. Integr.*, vol. 16, Dec. 2019, Art. no. 100106.

[11] Y. Du, C. W. de Silva, and D. Liu, "A multi-agent hybrid cognitive architecture with self-awareness for homecare robot," in *Proc. 9th Int. Conf. Comput. Sci. Educ.*, Aug. 2014, pp. 223–228.

[12] M. F. Pinto, L. M. Honório, A. L. M. Marcato, M. A. R. Dantas, A. G. Melo, M. Capretz, and C. Urdiales, "ARCog: An aerial robotics cognitive architecture," *Robotica*, vol. 39, no. 3, pp. 483–502, Mar. 2021.

[13] M. F. Pinto, A. L. M. Marcato, A. G. Melo, L. M. Honório, and C. Urdiales, "A framework for analyzing fog-cloud computing cooperation applied to information processing of UAVs," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–14, Jan. 2019.

[14] M. Seleckỳ, M. Rollo, P. Losiewicz, J. Reade, and N. Maida, "Framework for incremental development of complex unmanned aircraft systems," in *Proc. Integr. Commun., Navigat., Surveill. Conf. (ICNS)*, Apr. 2015, p. J3.

[15] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures and general intelligent systems," *AI Mag.*, vol. 30, no. 2, p. 33, 2009.

[16] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, and R. Singh, "Cognitive computing," *Commun. ACM*, vol. 54, no. 8, pp. 62–71, 2011.

[17] M. T. Cox, "Metacognition in computation: A selected research review," *Artif. Intell.*, vol. 169, no. 2, pp. 104–141, Dec. 2005.

[18] J. A. Reggia, "The rise of autonomous artificial cognitive systems: Issues and challenges," *IEEE Comput. Intell. Mag.*, vol. 8, no. 3, pp. 16–31, Jan. 2013.

[19] A. Chella, M. Cossentino, R. Pirrone, and A. Ruisi, "Robotic cognitive architectures: A perspective," in *Proc. Annu. Conf. North Amer. Fuzzy Inf. Process. Soc.*, Jun. 2009, pp. 1–6.

[20] D. Vernon, *Artificial Cognitive Systems: A Primer*. Cambridge, MA, USA: MIT Press, 2016.

[21] N. Hawes, "The STRANDS project: Long-term autonomy in everyday environments," *IEEE Robot. Autom. Mag.*, vol. 24, no. 3, pp. 146–156, Sep. 2017.

[22] J. L. Krichmar, "Design principles for biologically inspired cognitive robotics," *Biol. Cybern.*, vol. 112, nos. 1–2, pp. 97–111, 2018.

[23] G. S. Ramos, A. A. De Lima, L. F. Almeida, J. Lima, and M. F. Pinto, "Simulation and evaluation of deep learning autoencoders for image compression in multi-UAV network systems," in *Proc. Latin Amer. Robot. Symp. (LARS), Brazilian Symp. Robot. (SBR), Workshop Robot. Educ. (WRE)*, Oct. 2023, pp. 41–46.

[24] P. F. Dominey and F. Warneken, "The basis of shared intentions in human and robot cognition," *New Ideas Psychol.*, vol. 29, no. 3, pp. 260–274, Dec. 2011.

[25] P. Haazebroek, S. van Dantzig, and B. Hommel, "A computational model of perception and action for cognitive robotics," *Cognit. Process.*, vol. 12, no. 4, pp. 355–365, Nov. 2011.

[26] D. E. Rumelhart, "Schemata: The building blocks of cognition," in *Theoretical Issues in Reading Comprehension*. Evanston, IL, USA: Routledge, 2017, pp. 33–58.

[27] T. Ogata, K. Takahashi, T. Yamada, S. Murata, and K. Sasaki, "Machine learning for cognitive robotics," in *Cognitive Robotics*. Cambridge, MA, USA: MIT Press, 2022.

[28] M. F. Pinto, L. M. Honorio, A. Melo, and A. L. M. Marcato, "A robotic cognitive architecture for slope and dam inspections," *Sensors*, vol. 20, no. 16, p. 4579, Aug. 2020.

[29] F. A. de Alcantara Andrade, A. R. Hovenburg, L. N. de Lima, C. D. Rodin, T. A. Johansen, R. Storvold, C. A. M. Correia, and D. B. Haddad, "Autonomous unmanned aerial vehicles in search and rescue missions using real-time cooperative model predictive control," *Sensors*, vol. 19, no. 19, p. 4067, Sep. 2019.

[30] Y. Yang, C. Fermuller, and Y. Aloimonos, "A cognitive system for human manipulation action understanding," in *Proc. 2nd Annu. Conf. Adv. Cognit. Syst. (ACS)*, vol. 2, 2013, pp. 109–124.

[31] J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive robotics using the soar cognitive architecture," in *Proc. Workshops 26th AAAI Conf. Artif. Intell., Cogn. Robot.*, 2012, pp. 46–54.

[32] C. C. Insaurralde, "Service-oriented agent architecture for unmanned air vehicles," in *Proc. IEEE/AIAA 33rd Digit. Avionics Syst. Conf. (DASC)*, Oct. 2014, pp. 8B1-1–8B1-14.

[33] S. Emel'yanov, D. Makarov, A. I. Panov, and K. Yakovlev, "Multilayer cognitive architecture for UAV control," *Cognit. Syst. Res.*, vol. 39, pp. 58–72, Sep. 2016.

[34] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. Suárez Fernández, and P. Campoy, "A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework," *J. Intell. Robotic Syst.*, vol. 88, nos. 2–4, pp. 683–709, Dec. 2017.

[35] G. Asaamoning, P. Mendes, D. Rosário, and E. Cerqueira, "Drone swarms as networked control systems by integration of networking and computing," *Sensors*, vol. 21, no. 8, p. 2642, Apr. 2021.

[36] A. Gupta and S. K. Gupta, "UAV aided fog network (UAFN): A proposal framework for better QoS," in *Proc. 2nd Int. Conf. Comput. Inf. Technol. (ICCIT)*, Jan. 2022, pp. 265–270.

[37] H. A. Alharbi, B. A. Yosuf, M. Aldossary, J. Almutairi, and J. M. H. Elmirghani, "Energy efficient UAV-based service offloading over cloud-fog architectures," *IEEE Access*, vol. 10, pp. 89598–89613, 2022.

[38] Z. Chen, N. Xiao, and D. Han, "A multilevel mobile fog computing offloading model based on UAV-assisted and heterogeneous network," *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–11, Jul. 2020.

[39] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[40] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[41] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[42] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[43] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *IEEE Comput.*, vol. 49, no. 8, pp. 112–116, Aug. 2016.

[44] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[45] M. Satyanarayanan, "The emergence of edge computing," *IEEE Comput.*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[46] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Gener. Comput. Syst.*, vol. 70, pp. 59–63, May 2017.

[47] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[48] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed., MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[49] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. Australas. Telecommun. Netw. Appl. Conf. (ATNAC)*, Nov. 2014, pp. 117–122.

[50] J. Kuffner, "Cloud-enabled robots," *IEEE Internet Comput.*, vol. 14, no. 2, pp. 44–46, Jan. 2010.

[51] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[52] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[53] S. Sareen, S. K. Sood, and S. K. Gupta, "IoT-based cloud framework to control Ebola virus outbreak," *J. Ambient Intell. Humanized Comput.*, vol. 9, no. 3, pp. 459–476, Jun. 2018.

[54] O. Saha and P. Dasgupta, "A comprehensive survey of recent trends in cloud robotics architectures and applications," *Robotics*, vol. 7, no. 3, p. 47, 2018.

[55] T. Das, "Intelligent techniques in decision making: A survey," *Indian J. Sci. Technol.*, vol. 9, no. 12, pp. 1–6, 2016.

[56] S.-H. Liao, "Expert system methodologies and applications—A decade review from 1995 to 2004," *Expert Syst. Appl.*, vol. 28, no. 1, pp. 93–103, Jan. 2005.

[57] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[58] G. G. R. de Castro, M. F. Pinto, I. Z. Biundini, A. G. Melo, A. L. M. Marcato, and D. B. Haddad, "Dynamic path planning based on neural networks for aerial inspection," *J. Control, Autom. Electr. Syst.*, vol. 34, no. 1, pp. 85–105, Feb. 2023.

[59] S. Murshed, A. S. Nibir, M. A. Razzaque, P. Roy, A. Z. Elhendi, M. R. Hassan, and M. M. Hassan, "Weighted fair energy transfer in a UAV network: A multi-agent deep reinforcement learning approach," *Energy*, vol. 292, Apr. 2024, Art. no. 130527.

[60] S. Brotee, F. Kabir, M. A. Razzaque, P. Roy, M. Mamun-Or-Rashid, M. R. Hassan, and M. M. Hassan, "Optimizing UAV-UGV coalition operations: A hybrid clustering and multi-agent reinforcement learning approach for path planning in obstructed environment," *Ad Hoc Netw.*, vol. 160, Jul. 2024, Art. no. 103519.

[61] M. Leonetti, L. Iocchi, and P. Stone, "A synthesis of automated planning and reinforcement learning for efficient, robust decision-making," *Artif. Intell.*, vol. 241, pp. 103–130, Dec. 2016.

[62] T. D. Kelley, "Symbolic and sub-symbolic representations in computational models of human cognition: What can be learned from biology?" *Theory Psychol.*, vol. 13, no. 6, pp. 847–860, Dec. 2003.

[63] J. T. Ball, "Advantages of ACT-R over prolog for natural language analysis," in *Proc. 22nd Annu. Conf. Behav. Represent. Modeling Simulation*, 2013, pp. 53–60.

[64] M. V. Butz, "Toward a unified sub-symbolic computational theory of cognition," *Frontiers Psychol.*, vol. 7, p. 925, Jun. 2016.

[65] J. W. Tweedale, "A review of cognitive decision-making within future mission systems," *Proc. Comput. Sci.*, vol. 35, pp. 1043–1052, Jan. 2014.

[66] C. Sampedro, H. Bavle, J. L. Sanchez-Lopez, R. A. S. Fernández, A. Rodríguez-Ramos, M. Molina, and P. Campoy, "A flexible and dynamic mission planning architecture for UAV swarm coordination," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 355–363.

[67] M. A. Luna, M. S. A. Isaac, A. R. Ragab, P. Campoy, P. F. Peña, and M. Molina, "Fast multi-UAV path planning for optimal area coverage in aerial sensing applications," *Sensors*, vol. 22, no. 6, p. 2297, Mar. 2022.

[68] A. L. Alfeo, M. G. C. A. Cimino, N. D. Francesco, A. Lazzeri, M. Lega, and G. Vaglini, "Swarm coordination of mini-UAVs for target search using imperfect sensors," 2019, *arXiv:1901.02885*.

[69] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, vol. 3, Kobe, Japan, 2009, p. 5.

[70] Gazebo. (2021). *Gazebo Robot Simulator*. Accessed: Feb. 26, 2021. [Online]. Available: http://gazebosim.org/

[71] ROS. (2022). *Mavros*. Accessed: Dec. 4, 2022. [Online]. Available: http://wiki.ros.org/mavros

[72] MAVLINK. (2022). *Mavlink*. Accessed: Dec. 4, 2022. [Online]. Available: https://mavlink.io/en/

[73] PX4. (2022). *Px4 Autopilot*. Accessed: Dec. 4, 2022. [Online]. Available: https://px4.io/

[74] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, Havok, MuJoCo, ODE and PhysX," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 4397–4404.

[75] C. Anagnostopoulos, C. Koulamas, A. Lalos, and C. Stylios, "Open-source integrated simulation framework for cooperative autonomous vehicles," in *Proc. 11th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2022, pp. 1–4.

[76] OMNET++. (2022). *Omnet++: Discrete Event Simulator*. Accessed: Dec. 4, 2022. [Online]. Available: https://omnetpp.org/

[77] A. Mohini, "CDSSim–multi UAV communication and control simulation framework," M.S. thesis, Dept., Eng. Appl. Sci.: Comput. Eng., Univ. Cincinnati, Cincinnati, OH, USA, 2019.

[78] NS-3. (2022). *Ns-3 Network Simulator*. Accessed: Dec. 4, 2022. [Online]. Available: https://www.nsnam.org/

[79] PX4. (2021). *Controller Diagrams*. Accessed: Jun. 15, 2021. [Online]. Available: https://docs.px4.io/main/en/flight_stack/controller_diagrams.html

[80] PX4. (2021). *Multicopter Pid Tuning Guide (Manual/Advanced)*. Accessed: Jun. 15, 2021. [Online]. Available: https://docs.px4.io/main/en/config_mc/pid_tuning_guide_multicopter.html

[81] S. Acharya, A. Bharadwaj, Y. Simmhan, A. Gopalan, P. Parag, and H. Tyagi, "CORNET: A co-simulation middleware for robot networks," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 245–251.

[82] M. Calvo-Fullana, D. Mox, A. Pyattaev, J. Fink, V. Kumar, and A. Ribeiro, "ROS-NetSim: A framework for the integration of robotic and network simulators," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1120–1127, Apr. 2021.

[83] T. Liu, J. Wang, Q. Liu, S. Alibhai, T. Lu, and X. He, "High-ratio lossy compression: Exploring the autoencoder to compress scientific data," *IEEE Trans. Big Data*, vol. 9, no. 1, pp. 22–36, Feb. 2023.

[84] R. La Grassa, C. Re, G. Cremonese, and I. Gallo, "Hyperspectral data compression using fully convolutional autoencoder," *Remote Sens.*, vol. 14, no. 10, p. 2472, May 2022.

[85] S. Zebang and K. Sei-Ichiro, "Densely connected AutoEncoders for image compression," in *Proc. 2nd Int. Conf. Image Graph. Process.*, New York, NY, USA, Feb. 2019, pp. 78–83.

[86] S. M. H. Kalami. (2021). *Path-Planning: A Collection of Path Planning Algorithms*. Accessed: Mar. 24, 2023. [Online]. Available: https://github.com/smkalami/path-planning

[87] R. E. J. Kennedy, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov. 1995, pp. 1942–1948.

[88] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[89] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[90] S. M. LaValle, *Rapidly-Exploring Random Trees: A New Tool for Path Planning*, Standard TR 98-11, 1998.

[91] R. P. Foundation. *Raspberry Pi*. Accessed: Mar. 19, 2024. [Online]. Available: https://www.raspberrypi.org

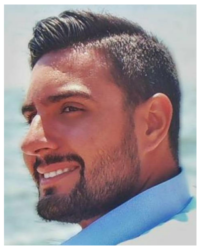
[92] I. Corporation. *Intel Galileo Board*. Accessed: Mar. 19, 2024. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/boards-kits/galileo.html

[93] G. S. Ramos. (Apr. 2024). *Arcog-Net—Simulation 01–Wind Energy Turbines Field Surveyllance*. Accessed: Apr. 17, 2024. [Online]. Available: https://youtu.be/LXOZyww0Jhs

GABRYEL S. RAMOS received the degree in electrical engineering from the Federal Institute of Education, Science and Technology, Campus Vitória, Espírito Santo, the master's degree in robotics from the Federal Center for Technological Education Celso Suckow da Fonseca (Cefet/RJ), and the Ph.D. degree in robotics from the Postgraduate Program of Instrumentation and Applied Optics (PPGIO), Cefet/RJ and UFF. He has developed work in the areas of robotics, instrumentation, artificial intelligence, digital image processing, and computer vision, working mainly on the following topics: mobile robotics, educational robotics, digital image processing, computer vision, AI, instrumentation, scientific instrumentation, and works in oil and gas sector, since 2018.

FELIPE DA R. HENRIQUES (Member, IEEE) received the degree in electrical engineering with an emphasis on telecommunications and the master's degree in electronic engineering from the State University of Rio de Janeiro, in 2006 and 2010, respectively, and the Ph.D. degree in electrical engineering from the Federal University of Rio of January, in 2015. He is currently the Director of Uned Petrópolis, CEFET/RJ, where he works as an EBTT Professor of technical courses in telecommunications integrated with the High School and computer engineering courses. In addition, he also works in the Graduate Program in Computer Science (PPCIC) and the Graduate Program in Instrumentation and Applied Optics (PPGIO), CEFET/RJ. He has experience in electrical engineering, with an emphasis on telecommunications systems, signal processing, and computer networks. He works mainly on the following topics: Wireless sensor networks, sparse representations and compressive sampling, and adaptive filtering.

DIEGO B. HADDAD (Member, IEEE) was born in Niterói, Rio de Janeiro, Brazil, in 1983. He received the B.Sc. degree in electrical engineering, in 2005, and the M.Sc. and D.Sc. degrees in electrical engineering from the Federal University of Rio de Janeiro, Brazil, in 2008 and 2013, respectively. He is currently with the Federal Center for Technological Education Celso Suckow da Fonseca (CEFET/RJ). His research interests include signal processing, machine learning, computer vision, and adaptive filtering algorithms.



FABIO A. A. ANDRADE (Senior Member, IEEE) received the Ph.D. degree in engineering cybernetics from Norwegian University of Science and Technology. He was an Adjunct Professor with the Federal Center for Technological Education Celso Suckow da Fonseca, Rio de Janeiro, and a Technical Advisor with TracSense. He was with Brazilian Department of Airspace Control and was an Assistant Professor with Brazilian Naval Academy. He is currently an Associate Professor with the University of South-Eastern Norway and a Research Scientist with the NORCE Norwegian Research Centre. He has several published works on robotics and machine learning. He is the Former Chair of the IEEE Norway Chapter of Robotics and Control Systems and the Co-Founder of Umaker.

MILENA F. PINTO (Member, IEEE) received the master's, Ph.D., and postdoctoral degrees in engineering electrical from the Federal University of Juiz de Fora (UFJF), with a doctoral year with Technische Universität München, and the degree in control and automation engineering from CEFET-MG. She was a Visiting Professor with the Polytechnic Institute of Bragança (IPB), Portugal. She was an Advisor for agreements and international relations with CEFET-RJ, from June 2022 to November 2022. She was the Coordinator of the Postgraduate Program PPGIO, CEFET-RJ/UFF, from February to June 2022. She is currently a Professor of the electronic engineering course (CCGELT) and a Permanent Professor with the Postgraduate Departments in Electrical Engineering (PPEEL) and Instrumentation and Applied Optics (PPGIO), Federal Center for Technological Education Celso Suckow da Fonseca (CEFET-RJ). She is a Young Scientist of Our State (FAPERJ). Her areas of research interests include artificial intelligence, embedded electronics, robotic systems, ROS, unmanned aerial vehicles, data processing, and fog-cloud computing.