Universidade Federal de Juiz de Fora

Faculdade de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

Milena Faria Pinto

ARCog: An Aerial Robotics Cognitive Architecture

Juiz de Fora

2019

Milena Faria Pinto

ARCog: An Aerial Robotics Cognitive Architecture

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora, na área de concentração em Sistemas de Energia, como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. André Luís Marques Marcato, Dr. (UFJF)

Coorientadora: Prof. Cristina Urdiales, Dr. (UMA)

Coorientador: Prof. Leonardo Mello de Honório, Dr. (UFJF)

Juiz de Fora

2019

Milena Faria Pinto

ARCog: An Aerial Robotics Cognitive Architecture

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora, na área de concentração em Sistemas de Energia, como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. André Luís Marques Marcato, Dr.
Orientador
Universidade Federal de Juiz de Fora

Prof. Cristina Urdiales, Dr.
Coorientadora
Universidad de Málaga

Prof. Leonardo de Mello Honório, Dr.
Coorientador
Universidade Federal de Juiz de Fora

Prof. Mário Antônio Ribeiro Dantas, Dr.
Universidade Federal de Juiz de Fora

Prof. Miriam Capretz, Dr.
Western University

Prof. Andre Gustavo Scolari Conceição, Dr.
Universidade Federal da Bahia

Prof. Eduardo Pestana de Aguiar, Dr.
Universidade Federal de Juiz de Fora

ACKNOWLEDGMENT

First, I would like to thank my family that for all these years has been my great supporters. I also would like to thank Aurélio, who, even living far away, have always demonstrated the meaning of fellowship, affection, and friendship.

An especial thanks to the professors Mário Dantas, Leonardo Honório and Cristina Urdiales for all support offered me during the development of my research. I also want to thank all professors from the postgraduate program at the Federal University of Juiz de Fora for making me understand not only what is to do research but also to make me appreciate the true meaning of doing science.

My special thanks to my advisor Andre Marcato for the opportunity to do this work, for believing in my research and for the wisdom in decision making. Thank you very much for believing in me.

“Creativity is contagious, pass it on”

Albert Einstein

RESUMO

A integração eficiente de algoritmos é uma questão fundamental na robótica aérea. No entanto, apenas algumas soluções de integração utilizam uma abordagem cognitiva. Abordagens cognitivas dividem problemas complexos em unidades independentes que podem lidar com interfaces de dados de nível progressivamente inferiores, por exemplo, sensores e atuadores, até dados de nível superior, tais como raciocínio e decisão. Uma arquitetura cognitiva define o fluxo de informações entre as unidades para produzir um comportamento inteligente emergente. Apesar das melhorias na tomada de decisão autônoma, várias questões-chave permanecem em aberto. Um problema é a seleção, coordenação e tomada de decisões relacionadas às diversas tarefas especializadas necessárias para o cumprimento dos objetivos da missão. Este trabalho de pesquisa aborda a tomada de decisão para uma arquitetura cognitiva usada em missões de Veículos Aéreos Não Tripulados (UAV), denominada ARCog, sigla de Arquitetura Cognitiva de Robótica Aérea. A arquitetura proposta estabelece as bases para o desenvolvimento de uma plataforma de *software* alinhada com os requisitos da tecnologia de ponta na área de pesquisa em questão. O sistema é projetado para fornecer decisões de alto nível. Além disso, a arquitetura permite a tomada de decisões em tempo real com comportamento social inteligente entre os agentes. Os principais algoritmos usados para testar a arquitetura em um ambiente externo são detalhados, bem como seus resultados. Os experimentos confirmaram o sucesso da arquitetura cognitiva proposta. Os resultados parciais mostraram a viabilidade técnica e a eficácia do ARCog em diferentes cenários, como em missões de Busca e Resgate, Inspeção e Vigilância.

Palavras-chave: Veículo Aéreo não Tripulado. Sistemas Inteligentes. Missão Autônoma. Arquitetura Descentralizada. Tomada de Decisão.

ABSTRACT

Efficient algorithm integration is a key issue in aerial robotics. However, only a few integration solutions rely on a cognitive approach. Cognitive approaches break down complex problems into independent units that may deal with progressively lower level data interfaces, e.g., sensors and actuators, up to higher level data, such as reasoning and decision. A cognitive architecture defines information flow among units to produce emergent intelligent behavior. Despite the improvements in autonomous decision making, several key issues remain open. A problem is a selection, coordination, and decision making related to the several specialized tasks required for fulfilling mission objectives. This research work addresses decision making for a cognitive architecture used to Unmanned Aerial Vehicle (UAV) missions, called ARCog, an acronym for Aerial Robotics Cognitive Architecture. The proposed architecture lays the groundwork for the development of a software platform aligned with the requirements of state-of-the-art technology in the field. The system is designed to provide high-level decision making. Besides, the architecture enables real-time decision-making with intelligent social behavior among the agents. The main algorithms used to test the architecture in an outdoor environment are detailed as well as their results. The experiments confirmed the success of the proposed cognitive architecture. The partial results have shown technical feasibility and effectiveness of the ARCog in different scenarios, such as Search and Rescue, Inspection and Surveillance missions.

Key-words: Unmanned Aerial Vehicle. Intelligent Systems. Autonomous Mission. Decentralized Architecture. Decision Making.

LIST OF ILLUSTRATIONS

ACRONYMS

ARCog	Aerial Robotics Cognitive Architecture
BOW	Bag-of-Words
CBR	Case-Based Reasoning
DoG	Difference of Gaussians
FCU	Flight Control Unit
FOV	Field of View
GPS	Global Positioning System
GUI	Graphical User Interface
HMI	Human Machine Interface
IMU	Inertial Measurement Unit
OpenCV	Open Source Computer Vision Library
PAL	Performance and Attention with Low-Task-Loading
QP	Quadratic Programming
ROS	Robotic Operating System
SAR	Search and Rescue
SDA	Surface Deformation Analysis
SDN	Software-Defined Network
SFM	Structure From Motion
SIFT	Scale-Invariant feature transform
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
VLOS	Vision Line of Sight
LF	Low Frequency

HF	High Frequency
UHF	Ultra High Frequency
TDD	Time Division Duplexing
FDD	Frequency Division Duplex
HSPA	High Speed Packet Access
FPGA	Field Programmable Gate Array

SUMMARY

1 INTRODUCTION

1.1 INITIAL CONSIDERATIONS AND PROBLEM CHARACTERIZATION

The vast number of applications using Unmanned Aerial Vehicles (UAVs), in recent years, has caused a rapid growth in research related to this technology, which is explained by the aircraft's ability to perform cost-effective activities with flexibility and reduced risks to human life. The demand for more complex missions has increased UAV capabilities. Besides, this demand generates platforms with a high degree of autonomy to carry out tasks simultaneously, with less human intervention [1]. The integration of complex algorithms and methods working on board the aircraft system has developed many highly specialized solutions to specific applications [2]. Solutions for certain tasks are usually limited by environmental understanding where specific algorithms can quickly achieve results. Examples of such a task include the Search and Rescue (SAR) mission, inspection, surveillance, activity recognition, monitoring, among others.

Many specialized systems are working on optimal solutions for specific problems. However, a few of those are integrated to advance the cognitive field and solve general issues underlying the aircraft's operation, such as decision making. Cognitive architecture organizes the way that the data and information of an agent flow, which is important because the structure supports the emergence of intelligent behavior. Cognitive research breaks the problem down and makes assumptions about the existence of low-level data interfaces, such as sensors and actuators. Besides, it should be focused on the information about those and about abstracting the hardware. The critical question comes in the designing of an architecture that can be used as a foundation where all different issues related to cognitive aspects can be represented, built, and solved incrementally. Examples of such problems are knowledge representation, decision making, and robotics.

Many works and methods have tackled the problem of defining decision making and knowledge representation from different angles. However, there are remaining improvements needed to advance this field of research. Several key issues remain open despite the improvements in this field. Therefore, this research work highlights the importance of addressing the development of practical methods for inserting artificial cognition in UAVs and mission goals oriented by decision making.

Robotic systems require a combination of several specific and interrelated building blocks to reach autonomous operation capabilities. Moreover, a strict organization is essential for interoperation and the emergence of intelligent behavior. However, this integration leads to new challenges, e.g., the adaptability for the execution of unique problems and scalability, among others. In this context, cognitive architectures are capable of mitigating those issues by speeding up development. An important aspect when comparing different methodologies is the criterion applied, which may vary according to the application. Some parameters can be used as a common point to evaluate the architectures. Examples are the ability to store and to represent knowledge, process natural language, decision making, problem-solving and the possibility of

real-time and real-world application.

An example of such a mission that cognitive architecture could improve is the reconstruction for inspection in 3D space. Visual dam inspection is a crucial safety task [3] that, if not properly executed, may have a tremendous impact in infrastructure and human lives [4, 5]. A critical part among all inspection categories is the correct instrumentation and monitoring system, which the main content is the 3D-dimensional surface deformation analysis (SDA). Usually, this activity is achieved by using 3D laser scanners [6]. However, this methodology is time-consuming and difficult to apply due to the size and complexity of some structures. In recent years, the development in UAV capabilities has dramatically changed this procedure.

More specifically about dam inspection using UAV, the literature shows several kinds of research. For instance, [7] presents potential applications of UAV along with photogrammetry to perform assessment and inspection of large structures. The approach generates points with sub-millimeter accuracy. From this work, it is possible to infer the benefits of using automated systems to perform inspection, such as simplified requirements. The work [8] uses a similar approach, where the images are also georeferenced by the application of markers in the inspected structure, increasing even further the reliability of inspections.

The work [9] exhibits an extreme application of UAV where the autonomous navigation system is applied to perform inspection inside Tunnel-like structure. This kind of scenario represents an edge case due to the complexity of the task performed by an autonomous system with minimal sensory input and reduced dimensions. This kind of survey would be complicated and risky for the human operator. Despite showcasing the technology potential, the autonomous navigation is very restricted to the specific task by combining laser scan and IMU to produce 3D localization and mapping features allowing the UAV to determine its position in the structure.

Considering the works mentioned above, the UAV executes a predefined task without any reactive regarding the quality of the inspection. Without loss of genericity, it is possible to state that any structure from motion (SFM) methodology based on photogrammetry needs to extract as much information from the image as possible. Therefore, if the UAV captures a series of pictures without the minimum amount of information, the final reconstruction may present a grotesque result. In such a situation, a change in the current path could improve the outcomes. However, the computational effort for a single UAV processor is too much to acquire and analyze an image, to evaluate the current point cloud-generation and to re-plan the mission along with low and high-level control functionalities. Therefore, an intelligent, scalable and distributed computational architecture is necessary if the task is scalable for a collaborative multi-agent team.

Another example of operation that can greatly benefit from the use of autonomous UAVs to survey the environment and collect evidence is the search and rescue (SAR) mission, which is often characterized by a similar set of constraints such as the time that can result in potentially human losses, areas with difficult access, e.g. forests, among others. UAVs can provide an

essential support to these operations once they are are fast, flexible and can exhibit autonomous behaviours.

In a typical SAR mission, the UAVs are deployed in a determined area to perform sensory operations, collect evidence of victims presence, and report a human operator for futher information or decision. UAVs with cognitive ability can potentially improve the rescue by taking actions to improve the certainty of the image recognition algorithm, intelligent data control to minimize network latency, reducing amount of operator required to control a team of aircraft, among others.

1.2 WORK MOTIVATION

Few works can be cited in the context of the learning process in autonomous systems [10], [11], [12], [13]. The system in [13] has a good ability to track but a low capacity to adapt and learn. The architecture proposed in [14] uses an automatic surveillance application for complex events recognition. However, differently, to the approach presented in this work, this methodology is based on ontologies and dedicated rule language using sensor data in association with simple action detectors to feed the reasoning engine responsible for the high-level formation. There is a similar hierarchical methodology in [15].However, there are requirements for specific sensor placement that turns more difficult the practical application in outdoor environments.

An essential contribution of this work, coined as ARCog, is the definition of the components of an aerial robotic cognitive architecture. The proposed architecture organization lays the groundwork for the development of a software platform aligned with the current requirements of state-of-the-art technology, such as semi-autonomous navigation capability, environment perception and decision. The focus on cognition means that the system is designed to be capable of high-level decision making, such as understanding, interpretation and goal-oriented reasoning. The most important characteristics for a proper autonomous UAV operation in a real-world environment are the low response time for decision making, predictability to assure safe operation, learning capability, high flexibility and adaptability for operations.

This research proposes an architecture based on the biologic model of the human cognitive system consisting of perceptual and reasoning modules to achieve these desired characteristics. An important aspect to be obtained is the development of a flexible architecture because most of the employed algorithms and methods may improve over time and the proposed system will be able to absorb most of the innovations that can arise in future without unnecessary changes in the whole structure. Note that several aerial applications are mostly based on centralized control, where the processing is carried out in a central computer [16], which is a not good scenario once network delays or failures can cause the system to stop searching or lose an important event that may happen. This work proposes a decentralized application, which gives rise to another technological limitation present in the current technical state, i.e., the computational capabilities of current embedded computers that limit the algorithm's selection. Therefore, this research aims

at the development a low power processing architecture to be inserted in a companion computer onboard the UAV.

Most works using autonomous UAVs are evaluated only in simulation and lacks in the validation of practical applications. In this way, in the presented approach, the activities in the missions are recognized and monitored by an UAV, and the experiments were designed to contemplate possible real-world scenarios. The system should be capable of evaluating effectively a sequence of the sub-actions present in each scene provided by similar and past problems.

1.3 MAIN CONTRIBUTIONS

The aerial application requires algorithms that are consistent and predictable to allow proper verification and validation. The ARCog captures the current state-of-the-art in technologies and algorithms, Human-in-the-loop (HITL) and team coordination for autonomous operation while being also human-friendly. This methodology uses language to represent the information and for the decision process. A functional block in the high-level uses the case based reasoning method [17] to learn and to perform decision making. The architecture is in a Linux compatible hardware, and it works in as close as real-time as possible with data processing onboard the aircraft. The ARCog presents an easy interchange between algorithms and methods for a hardware interdependence.

Therefore, the main contributions of this research are divided among application, architecture, and implementation, as follows.

- **Development of a differentiated architecture to support decision making and emergence of intelligent behavior for UAVs**

  The architecture flexibility enables the update of most of the employed algorithms and methods without requiring change in the whole architecture. Note that algorithms for embedded applications are especially tricky cases due to the processing capability of the available hardware. Therefore, there is a lack in the current technologies to execute the most advanced methods. Thus, as they can improve over time, the proposed system will be able to absorb most of the innovations.

  The organization of the ARCog is a central point to achieve the proposed objectives due to the architecture's capability of improving understanding and boosting performance. A common characteristic observed in different propositions found in the literature is the component organization, i.e., the blocks are inter-organized to match functional aspects, such as hardware interface, control loops, planning, among others. The organization eases the visualization of the component's functionalities. Thus, with the ARCog organization, there is no need to rewrite the entire code due to architecture dynamics. For example,

changes in the low-level do not interfere in the other levels, such as in case of a simple microprocessor be able to run onboard the deep learning algorithm.

- **A proposition of a model to analyze fog-cloud computing cooperation applied to information processing to a system that operates in a decentralized and autonomous way with centralized coordination and teamwork.**

  This research proposes a model for analyzing the feasibility of fog-cloud computing cooperation for UAVs. The primary goal of this model is to improve the throughput and latency limitations in a mission involving multiple aircraft. In this architecture, the aircraft assigned as a head coordinator handles the communication between the nearest node and the ground station that is located in the cloud. The head coordinator UAV will be responsible for managing the computational offloading between cloud and fog.

  The advantage of a decentralizing operation is that in case of communication loss, the system can record the images downloading them after the landing or make autonomous decisions in a failure case.

- **A new grammar-based system based on the well-known weighted automata.**

  This approach is based on deterministic finite automata (DFA) [18]. The primary goal is to make the aircraft perception system (i.e., the vision system) be related directly to inside knowledge of the world model. A specific grammar rule runs the perceptual signals obtained through the computer vision and the events identification. For example, in a surveillance mission, this grammar rule would be SUBJECT-ACTION-OBJECT and LOCAL + TIME. As an example, the grammar can be composed of Person (Subject) – Holds (Action) – Gun (Object) – Parking lot (Place) – Work hours (Time). The grammar governs these signals.

1.4 RELATED PUBLICATIONS

The presented research work has resulted so far in publications in two international conferences, in a Brazilian conference and an article published in a journal. The related publications are listed below.

- “Case-Based Reasoning Approach Applied to Surveillance System Using an Autonomous Unmanned Aerial Vehicle”, M. F. PINTO, A. G. MELO, A. L. M. MARCATO, C. URDIALES, in 26th IEEE International Symposium on Industrial Electronics, Scotland, 2017.

- “Design for Online Trajectory Prediction of a Moving Object Detected Onboard of a UAV”, M. F. PINTO, F. O. COELHO, J.P.C. SOUZA, A. G. MELO, A. L. M. MARCATO, C. URDIALES, in CONTROLO, Portugal, 2018.

- “Remoção Dinâmica de Plano de Fundo em Imagens Aéreas em Movimento”. MILENA F. PINTO; AURELIO G. MELO; CRISTINA URDIALES; ANDRE L. M. MARCATO (2018). 2018 XXII Congresso Brasileiro de Automática (CBA).

- “A Framework for Analyzing Fog-Cloud Computing Cooperation Applied to Information Processing of UAVs”. MILENA F. PINTO; ANDRE L. M. MARCATO; AURÉLIO G. MELO; LEONARDO M. HONÓRIO; CRISTINA URDIALES (2019). Wireless Communications and Mobile Computing. Volume 2019, 1-14, 2019. https://doi.org/10.1155/2019/7497924

The first paper, Case-Based Reasoning Approach Applied to Surveillance System Using an Autonomous Unmanned Aerial Vehicle, consists of the essential basis that led to the development of this thesis with a concept of a three-layer hierarchical architecture. This work also includes the study of the CBR algorithm as part of the high-level information processing. This algorithm takes the natural language output coming from the automata and from the sensors data to produce a partial view of the UAV world. The incomplete scene is compared against the cases in the database to predict UAV behaviors and to evaluate the correctness of the current path planning.

The second paper, Design for Online Trajectory Prediction of a Moving Object Detected Onboard of a UAV, performs a study and implementation of a methodology for object position recognition concerning real-world coordinates while discarding the UAV movement. Besides, this work provides the results of the image recognition algorithm in the detection of human suspicious threat activities in real-time surveillance applications.

The third paper, Remoção Dinâmica de Plano de Fundo em Imagens Aéreas em Movimento, consists of a dynamic removal of backgrounds of aerial images. The background subtraction technique is used to reduce the amount of data in each image by removing the foreground of an image and leaving only the regions of interest. Most works that perform background removal in areal images assumes scenarios with a static background. The methodology proposed in this paper determines the image movement and corrects the foreground data position to allow background removal of moving cameras.

The last published article, A Framework for Analyzing Fog-Cloud Computing Cooperation Applied to Information Processing of UAVs, tackles the problem of information exchanged among multiple aircraft and between aircraft and ground station. The exchange of information can be limited by high distances and low bandwidth besides being restricted by the processing capability, and energy constraints. This article presented a mathematical model to analyze distribution-based UAVs topologies and a fog-cloud computing framework for large-scale mission and search operations.

1.5 WORK STRUCTURE

In addition to this introductory chapter, this thesis is divided into six more chapters. Chapter 2 presents a brief review of the related works highlighting the state-of-the-art cognitive architectures. Besides, this chapter discusses the advantages and constraints that can be optimized when applying fog computing to work cooperatively with cloud data centers in a UAV scenario.

Chapter 3 describes the proposed innovative multi-level architecture ARCog and the selected methods used to implement the problem solution. This chapter also introduces an innovative approach for the well-known weighted automata, which propagates the certainty associated in each recognized instance to the high-level decision maker.

Chapter 4 details the problem formulation of the proposed fog-cloud computing framework and the methodology to evaluate its applicability for UAVs aiming at optimizing latency while keeping throughput and power consumption under a range.

Chapter 5 contains the experiments and the respective results obtained through the application of the ARCog for SAR, surveillance and inspection missions. This chapter also presents a proper discussion of the results.

Chapter 6 exhibits the conclusions concerning the results obtained through the application of the ARCog in different scenarios showing its robustness and effectiveness. Extensions for the ARCog and ideas of improvements are also shown in this chapter.

2 BACKGROUND AND RELATED WORK

With the advent of new technologies, such as power processing boards and sensors, UAVs have become increasingly popular. One of the main reasons is that they can cover a larger area and they are cost-effective. For instance, UAVs can bring more efficiency to missions like surveillance due to fatigue that may cause to the human operator when operating multiple screens for several hours. Autonomous aerial systems often use artificial intelligence for comprehending scenes and computer vision techniques to recognize specific actions. Therefore, in this chapter, we briefly introduce the generic aspects of cognitive architectures and aerial robotic systems frameworks. This chapter also presents a brief overview of the algorithms used for building the ARCog.

2.1 COGNITIVE ASPECTS

According to psychology, cognition is the knowledge learning process that occurs through perception, attention, association, memory, reasoning, judgment, imagination, thought and language [19]. Cognition is a conversion mechanism of what is obtained by the interaction with the environment, capturing and processing the stimuli selected of the internal mode. In simple words, it is possible to say that is the way that the brain perceives, learns, remembers and thinks about information received through sensors.

Language acquisition is part of human cognitive development, and humans acquire language naturally. They use for different reasons, such as information, communication, storage and transmission [20]. There are a variety of approaches to understanding language acquisition. An important theory comes from the linguist Noam Chomsky and Lightfoot, which stated that humans are all born with an innate grammar knowledge that serves as the basis for language acquisition and in this way, the language process is a basic instinct inherent to the human being [21]. Chomsky's and Lightfoot contribution was central to the language computational analysis. The work also developed structures and syntactic operations. Grammars contribute to providing syntax and semantics of common actions and in this way, basic tools for understanding it.

There are a significant number of works proposing cognitive architectures, such as in [22] and in [23]. This first one aimed at solving advanced cognitive tasks and the second one to help the interaction between the robotic system and objects. Similar to other cognitive architectures, this research considers that the robotic system has appropriate sensors units and an artificial cognitive system containing a simple flow of information among modules for generating a reasonable behavior when performing tasks. Based on these works, the contributions of the proposed cognitive architecture cover three problems:

- Simple information flow between perceptual modules and high-level inference layer;
- Relevant information of perception-action;

- Activities composed by sub-actions to ease the organization and complex tasks interpretation.

A relevant feature in cognitive systems is the data representation. A dichotomy arises in the aspect of symbolic and sub-symbolic paradigms [24]. The first one uses direct symbols for data representation such as words and ontologies. The symbolic representation has a data processing block that is responsible for accepting symbols and for producing an output. The sub-symbolic paradigm uses a parallel data representation that is not directly accessible, such as the weights in a neural network. Hence, symbolic representation is the most appropriate for aircraft applications due to the possibility of verifying the represented knowledge. Besides, the symbolic paradigm allows human debugging and a better understanding of decisions.

In this aspect, the presented generic architectures SOAR [25] and ACT-R [26] do not define the knowledge representation in a specific way in their original propositions. Subsequent works have published structures capable of such representations, i.e., either symbolic [27] or sub-symbolic [28]. These evaluated systems are well defined, and they use symbolic knowledge. In this research work the exchange among blocks will be primarily performed through symbolic representation.

Another important feature during the interaction between the system and the operator is the aptitude for natural language representation. There are extensions to add these features in the general propositions. For example, NL-SOAR [29] is an improvement of the SOAR that can comprehend and generate outputs in natural language. This extension can be used in both simulated and real-time environments. Another equivalent improvement is in the architecture CHREST [27] that was employed in the work of Insurralde et al. [30]. In Ball et al. [26], the proposition ACT-R was designed to support natural language. The architecture proposed in this research uses symbolic knowledge for data exchange among the main blocks. These implementations support symbolic processing. All required non-symbolic methods such as computer vision are enclosed to result in symbolic outputs. Thus, the reasoning performed within the architecture can be fully understood while analyzing the data flow.

The next important topic is how to use data representation for problem-solving and decision making, which is extremely necessary once a fully autonomous behavior requires an operation in a dynamic environment and not all decisions can be directly programmed in the system memory. Another relevant topic is the ability to learn through previous experience. The capability of the system to learn from its experience can further improve its aptitude for responding to future events. Besides, this reduces the work required at the programming stage.

Tweedale [29] presents a review of the generic methodology employed by SOAR and ACT-R in the decision-making process. This work uses expert knowledge and situational memory combined with rule-based logic to produce reasoning, which is similar to the method presented in Seleckỳ et al. [1]. The methodology of [31] uses a heuristic approach to perform fast movement

planning. The concept of planning and decision making are in the propositions of Seleckỳ et al. [1] and Insaurralde et al. [30], but there is not an implementation description. The work of Sampedro et al. [32] shows an implementation where a centralized activity manager is responsible for optimal tasks distribution. The agent has an onboard decision-making algorithm responsible for path planning. However, this agent is not prepared for decision making in case of broader and general missions.

2.2 AERIAL ROBOTIC SYSTEMS FRAMEWORK

High-level decisions in SAR applications are related to target detection, recognition [33], [34] and team coordination [35]. However, more generic decision making is not the usual scope in this context. However, more generic decision making is not the usual scope in this context. For instance, the presence of a decision-maker enables the operation of the UAV, in the most dynamic environments even without central planning or management. The aircraft can respond optimally to any environmental changes.

There are many proposals creating approaches for intelligent systems construction. The Aerostack [31] is one of the most prominent solutions. This work presents a modular organization for command and control of autonomous, heterogeneous teams. Moreover, this approach provides an incremental development and verification of a complex system using a mixed-reality concept. The operation is conducted onboard the aircraft. The tasks are path planning and collision avoidance. As a drawback, this model is centered on sensorial data, and it does not define a specific model for data storage. Besides, the architecture leaves a blank for improvements in HITL, team coordination, and autonomous decision making.

The proposed Intelligent Vehicle Control Architecture (IVCA) [36] enables collaboration among multiple air vehicles. The IVCA is generic, but it aims at the study of aircraft working cooperatively. This work is based on ontologies, and it does not contain specification about hardware nor any explanation of learning capability. Seleckỳ et al. [1] shows a fully UAV an architecture for autonomous operation centered more on sensors data. However, the learning capability is not implemented, and the hardware requirements depend on the work extension.

Based on the presented developments, ARCog is comprised of perceptual, structural and reasoning levels in order to develop a autonomous system. This architecture captures the current state-of-the-art in technologies, algorithms, HITL and team coordination for autonomous operation whilst also being human-friendly. This architeture uses language to represent the information and for decision making. A specific high-level block uses a method to learn and for decision making. The architecture works with data processing aboard an aircraft in real time. The architectural approach of using blocks presents a model for easy change between different algorithms, by replacing the specific block. The usage of ROS also allows for FCU hardware interdependence once any ROS compatible FCU can be used. Table 1 presents some of the architectural differences among related architectures as well as the highlighted advantages of the

ARCog.

Table 1 – Features of some related aerial robotics architectures.

Features	**Aerial Robotics Architectures**					
	ARCog	FFIDUAS [10]	IVCA [12]	STRL [17]	Proactive [40]	Aerostack [11]
Practical experiments showing architecture operation.	Yes	Yes	No	Yes	No	Yes
Multi-agent support	Yes, it applies fog to coordinate the agents	Yes, it is centralized on the cloud	Yes, however does not define the control structured	No, only a single agent design is presented	Yes, it is centralized on the Cloud	Yes, but it does not define explicit coordination among agents.
User decision support	Yes	No	No	No	No	No
Can incorporate human experiential knowledge into the model	Yes, CAD can learn users decisions	No, does not define this feature	No, does not define this feature	No, does not define this feature	No, does not define this feature	No, does not define this feature

2.3 FOG COMPUTING

In regular missions, the operator controls the vehicle position in every situation. However, when the mission is semiautonomous, and the operator is responsible for just a few tasks, such as taking off and landing the aircraft while this one carries out the flight autonomously through waypoints. In fully autonomous reactive missions, the trajectory is created onboard the aircraft and the mission is performed without the operator nearby. In this sense, the autonomous aerial robotics are connected to a supervision system (i.e., ground station—GS) that is usually located at the cloud and is responsible for all high-level processing.

It may be ineffective to offload an extraordinary amount of data to the cloud due to the high cost of communication bandwidth, energy constraints of embedded systems and redundancy of sensor data [37]. Instead of offloading data to the cloud, it may be efficient to allocate computational tasks closer to the end-devices intelligently. Cloud computation faces several challenges including lack of mobility for data processing and delay associated with network

transmission. This delay varies due to numerous factors such as accessing methods (i.e., 3G, 4G, Wi-Fi, among others), the number of hops from sources to the GS and signal interference in the area.

This cloud-based approach may be inappropriate for sensitive real-time systems once the exchanged amount of data among the aircraft would generate higher costs of communication bandwidth, lack of mobility, communication delay, energy constraints of embedded systems, and information redundancy. To mitigate these issues, a new trend of computing paradigm is to make the computation and storage close to the end-devices, which in this particular case are the drones. Fog computing arises as an intermediate layer between cloud and end-devices to improve latency, power consumption, scalability, and efficiency. This technology allows overcoming the limitations of centralized cloud computation by enabling data acquisition, processing, and storage at fog devices [38]-[39].

Therefore, the architecture proposed in this research uses a structure based on fog computing to overcome throughput and latency limitations. This work considers that one UAV is assigned as a head coordinator is in charge of the fog-cloud computational offloading. The head coordinator analyzes the incoming data and chooses to process locally or to send it to the cloud. The information that goes to the cloud is grouped into chunks before being transmitted.

Several pieces of researches have been published to formally define the fog computing with its respective challenges. Its benefits and issues are surveyed in Dastjerdi and Buyya [40] and Mouradian et al. [41] by presenting an overview of this topic along with its characteristics. Some discussion of challenges, application scenarios and emerging trends can be found in [42], [43] and [44], respectively. However, there is a difficult to use the available fog platforms in remote areas due to an unreliable connection. A practical application of fog computing requires an architecture to achieve the proposed goals. For instance, the work [45] presents a fog architecture with a flexible software-defined network (SDN) to programmatically control networks. The devices in this network should present a flexible self-organized structure to allow the insertion of fog nodes. [46] introduced the concept of the virtualization of services where a fog implementation based on ROS performs the services for a network of robots. This implementation resembles the one proposed in this research work.

A few other topics related to the fog application in this context are worth mentioning. An implementation to minimize the services delay is presented in [47]. This work proposes a policy for fog nodes considering queue length and different request types with variant processing times. Other active topics are the complex requirements to obtain a highly reconfigurable network [48], and the paradigm of implementing shared services and resources [49], [50]. These subjects are essential to the UAV-Fog operation but are not discussed in the present work.

This fog-cloud approach is added to the ARCog architecture to optimize three subproblems. The first and most important one is the latency perceived by the end-user (i.e., UAV). Currently, in several applications, the interaction between a fleet of UAVs and cloud is performed

individually. However, this may be inefficient and costly if the volume of data increases, which may also present high redundancy.

The latency is modeled in slightly different ways in the literature. In [51], the latency is calculated by combining the time required for transfer data between nodes, the processing time, and the period related to balance the services among nodes. In [52], these previous factors are considered along with a nonlinear component that accounts for differences in transmission channels such as the queuing order along nodes. Other models may also consider factors like average transmission error in the networks, inter-UAV communication latency, and the time required for clustering data in fog [53], [54]. There are also different modeling techniques which include statistical analysis and modeling based on queue theory where the latency is calculated as the average response time in a queue model M/M/1 [55]. The M/M/1 represents the queue length in a system having a single server, where arrivals are determined by a Poisson process and job service times have an exponential distribution.

The second considered characteristic is the throughput between fog and cloud computing. The throughput may be affected by some factors, including limitations in the hardware, available power processing, and end-user behavior. Despite being a challenge for fog application, these factors are usually not modeled. Throughout the literature this is compressed into a single rate of error in the transmission channel [56]. In [57] and [58], the throughput of 3G and 4G networks is analyzed as a network peak data ratio distributed among connected UAVs.

The last analyzed characteristic is power consumption. The work [59] presents a model to analyze the power consumption and to evaluate the tradeoff between power usage at fog and cloud nodes during the network operation. The model considers the fog computing as a data center. This consideration provides preloaded content to end-users. Despite the good results, the model looks at the problem to save energy at the cloud servers. In this proposed research, the problem restriction is to optimize power consumption at the fog devices to extend the UAV flight time. The power consumption of data-forwarding is analyzed in [51] and [55]. The fog device model is seen as a resource with unlimited access to power supply, which is not real for an embedded application. In this sense, the current state of the art can be improved with models that reduce latency while still maintaining an optimal power consumption in the fog level.

2.4 MAIN ALGORITHMS USED IN THE ARCog

Several specialized methods are required to build an autonomous aerial architecture for inspection, SAR and other kinds of missions. Despite the complex construction, different processing levels lead to one advantage, i.e., the ability to organize and to scale the software development of all those methods. Each level of the ARCog should contain an interface to transfer data to the next stage as independently as possible. Note that the methods choice heavily influences the overall system performance. Thus, the following subtopics detail a brief discussion and the justification of the methods selection that are:

- **Layers 2 and 3**

  Lucas-Kanade [60] and Difference of Gaussian (DoG) methods to perform background subtraction; Bag-of-Words (BOW) [61] for object detection and recognition; Modified Automata [18] for structuring the information; CBR [17] for inference over the information and for organizing the data from several UAVs (e.g., sensory data, detected actors, objects and actions) along with the operator past decisions;

- **Layer 4**

  HMI to interface the communication between human operator and aircraft.

2.4.1 COMPUTER VISION

Computer vision methods allow the understanding of surroundings by robotic systems in the same way that humans do. Computer vision is comprised of several steps that include acquisition, processing and analyzing images to enable their usage by artificial intelligence systems. The computer vision algorithms include the background removal, classification and recognition of objects in the images, and object position identification w.r.t. image and earth frames.

2.4.1.1 BACKGROUND REMOVAL

The computational load for the recognition system can be directly linked to the number of processed pixels. Note that the reduction of the input image number of pixels can increase processing speed. However, there is a tradeoff between image resolution and the minimum amount of details to allow proper object recognition. Note that the UAV tends to fly as high as possible to avoid obstacles. Thus, this decreases the image resolution, and it may cause blur in the entities that the computer vision algorithm wants to detect. Another way to reduce the number of pixels is to remove the background once the image captured by the UAV contains more background than moving entities.

Background removal consists of subtracting the static parts of a given set of images from new income ones. In presence of a motion between images, the standard background removal methodology becomes impractical due to the difficulty to determine the background model. A simple solution is to match the subsequent images positions with the common parts of previous images to overcome this problem. Then, the algorithm needs to learn the new parts of the background model.

The images movement needs to be accurately determined to implement the proposed background removal methodology, which can be performed using optical flow methods. Currently, there are more than thirteen methods scientifically published and evaluated [62]. This proposed architecture applies the well-known Lucas-Kanade method [60]. The technique is part

of the native OpenCV library presenting a good relationship between performance and accuracy. Note that this is not a critical part of this research.

Figure 1 shows a simplified diagram of the background algorithm. The process consists of determining the image features and their tracking across image sequences. The features displacement between two images provide a good indication of the image movement, and can be used to compute the homography. Then, the background model is initialized using the first image, and a learning rate in the algorithm ensures a proper update of the model. When the system captures the incoming image, the optical flow method determines the movement of the features, and this incoming image is positioned by homography to the new position. The algorithm removes the background of this image to the next processing stage. The original image is feed into the background model. Note that some critical characteristics arise in this process. First, if an entity is static in the same position for a few frames, its presence is hidden once the algorithm will consider the entity as part of the background during these updates. Second, the model is only appropriately updated if the changes among the images are gradual enough to allow a suitable overlap and feature tracking. Third, the homography does not assume any direction of the movement and can also be used with the camera rotation. Besides, if the UAV altitude changes, this is perceived as image scaling.

Figure 2 depicts the two mains steps of background correction. The first one calculates the optical flow between two windows using Lukas-Kanade [60]. This method provides an offset vector that represents the motion. The second one produces the homography of the subsequent image by positioning it in a way that the two images correspond. As a result, each subsequent window has matching features that enable the background removal application. Note that new parts are added continuously in the image in the direction of the UAV movement, creating the need for constant background learning in each window with enough gain to learn quickly new static parts. This process should be following the expected movement speed of the aircraft.

Figure 1 – Background Subtraction Algorithm.

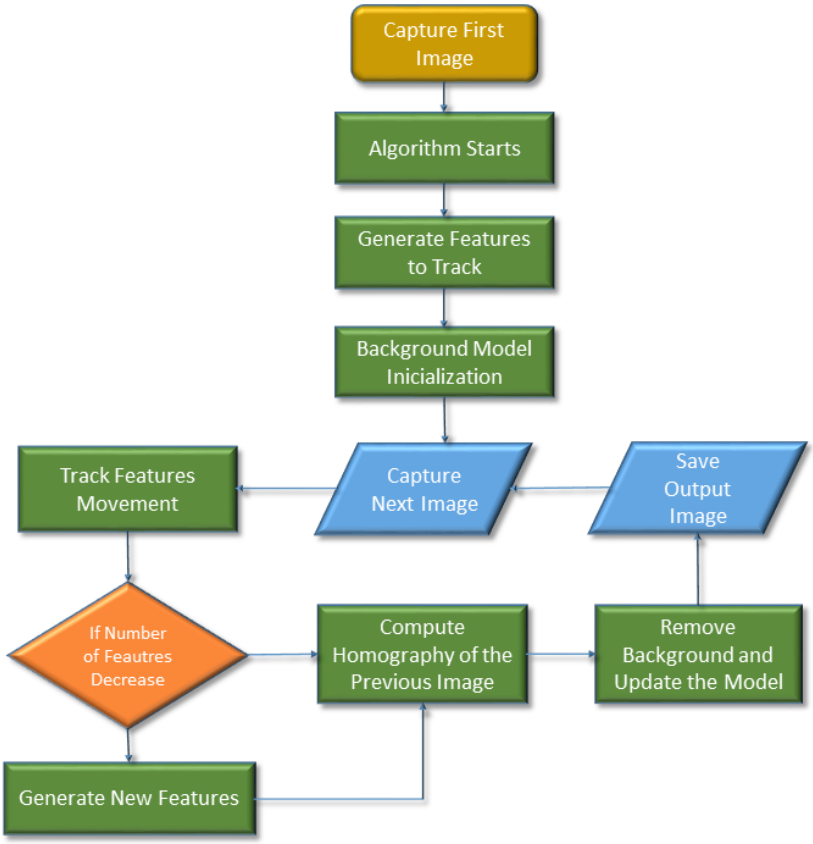
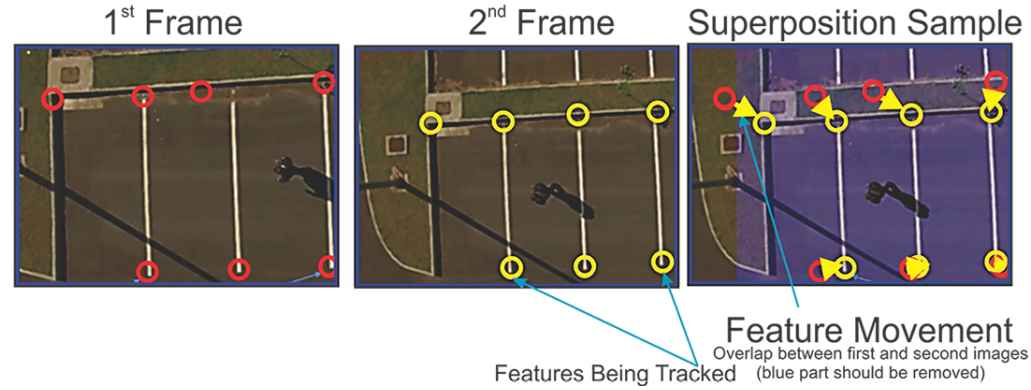


Figure 2 – New Window Correction.



After this process, the object recognition starts. The clusters are submitted to a trained classifier that determines which classes are present in the image. Then, this algorithm builds a vocabulary of visual words. The classifier is the Support Vector Machine (SVM). The SVM has the goal of learning the differences in each image class [63]. The SVM is a classifier based on statistic learning theory. It has an advantage of handling large sets of examples, high-dimensional data, and the classification is a fast process. Firstly, it is necessary to name a few symbols to state the SVM. Considering a weighted vector ω and the bias β, the equation 2.1 defines a hyperplane.

The variable x represents the training examples closest to the hyperplane. A signal function g(x)=sgn(f(x)) can be used to obtain the classification.

$$f(x) = \omega \cdot x + \beta \tag{2.1}$$

The distance from the hyperplane to a determined reference data point (i.e. margin) is defined as in 2.2. Note that the closest distance to the element is at least 1.

$$dist = \frac{1}{\| \omega \|} \tag{2.2}$$

Note the separation margin maximization of the data in relation to 2.1 is obtained by the minimization of $\| \omega \|$. Thus, the optimization problem relies in finding the minimum of 2.3 subject to the constraints $\omega \cdot x + \beta > 1 \forall i = 1, \ldots, n$.

$$\min_{\omega,\beta} \frac{1}{2 \cdot \| \omega \|^2} \tag{2.3}$$

2.4.1.2 OBJECT RECOGNITION AND CLASSIFICATION

Object recognition is a branch within this topic that describes image processing with the goal to detect, localize and classify instances in the image. There is a vast number of approaches with different goals and characteristics. The BOW method, proposed by [64], extracts a set of images features, and it uses a classifier to determine the presence of an object given a set of features. The BOW descriptors are mathematical functions responsible for creating the same output when subjected to similar images. SIFT and SURF are among the most known descriptors. This research uses the SURF descriptor to compute the key points due to the reason that this method is invariant to image scale and rotation as well as robustness to affine distortion and change in 3D viewpoint [65]. These descriptors are clustered and classified to determine which set of features belong to each object class. If a database containing enough examples of each class is used to train a given classifier, the algorithm is capable of deciding if a particular key points group contains an object. After this process, an attention mechanism (i.e., sliding window) is applied over the desired image to search for the desired entities and the object position in the image.

A few examples can be cited to illustrate the utilization of BOW in modern applications. The work presented in [66] uses this algorithm to detect flying objects, such as birds and other UAVs with accuracy up to 80%. The work of [67] applies BOW along with a new proposed methodology to segment roads, and to perform cars classification present in the images with accuracy up to 90%. In the field of robotics, there is also modern applications such as in [61] where the BOW performs robust visual objects tracking. The results are similar to the literature state-of-the-art.

Another object recognition algorithm is the Principal Component Analysis (PCA). This method compiles the training dataset to produce a subspace namely eigenspace, and the input image windows are projected in this subspace to be compared to determine the presence of the object [68]. The artificial neural networks, e.g., Convolutional Neural Networks (CNN), has several layers where each one of these extracts a group of features that are reduced (i.e., layer by layer) until obtaining a single dimensional output vector representing the classes at the input image.

Table 2 compile the characteristics of relevant algorithms used in recognition. The rating in the table is based on the results in the reference by splitting the results into 5 linear categories. Very low is results between (0 - 20%), low (20 - 40%), regular (40 - 60%), high (60 - 80%) and very high (80 - 100%). According to [69], [70] and [71], the two different ways to extract features using BOW method are the Scale-Invariant Feature Transform (SIFT) [72] and Speeded Up Robust Feature (SURF) [73]. Note that despite the high average accuracy of CNN based methods, they are not as fast or easy to compute as the BOW method. These are the main constraints factors to the application of UAV in real-time missions, which requires fast computational speed. The available embedded computers do not have a huge amount of processing power to enable high demanding algorithms. The best balance is given by the BOW method using either SIFT or SURF descriptors.

Table 2 – Main Characteristics of the algorithms BOW (using SIFT and SURF Descriptors), PCA and CNN for object recognition [69], [70] and [71].

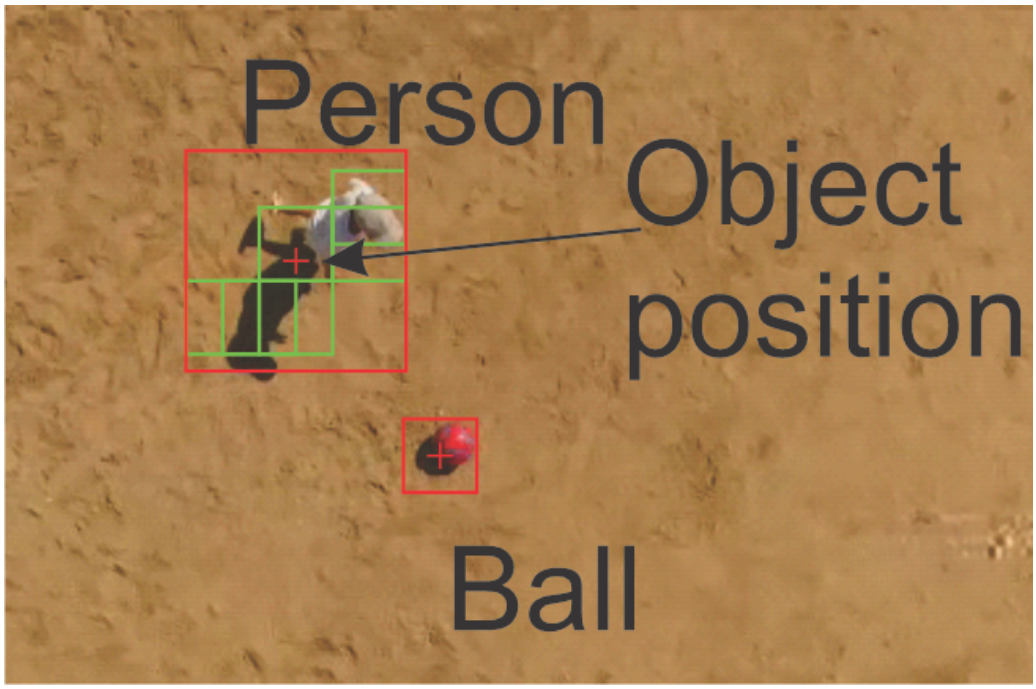
	SIFT	**SURF**	**PCA**	**CNN**
Invariance to Rotation	High	High	Regular	High
Invariance to Illumination	Neutral	High	Low	High
Invariance to Occlusion	High	Regular	Low	High
Speed	Regular	High	Neutral	Low
Accuracy (average)	Regular	High	Regular	Very High
Algorithm Complexity	Regular	Regular	Low	High
Computational Load	Regular	Regular	Regular	High

2.4.2 OBJECT POSITION IDENTIFICATION

Only a small fraction of the information recorded by the visual system reaches relevant processing levels that directly influence the behavior [74]. The attention mechanism is employed to optimize the ability to locate entities in a scene. This engine searches for objects in more details areas where there is a presence of desired objects and performs a less detailed search in free zones. The first scan of this attention mechanism is standard, e.g., it moves 30 pixels at a time. However, if an entity has been detected, the sliding window decreases the step in the subsequent scans over this area, e.g., moving 10 pixels at a time. This engine allows great object detection without overloading the algorithm and maintains a minimum number of frames per second.

In the sliding window process, the original image is sliced in windows, and the presence of the entity is individually searched in each frame as proposed in [75]. If the SVM determines the object features in a window, the entity position is associated with the center of this window. These two processing steps (i.e., entities presence recognition and position determination) may be slow if the number of known entities on the software database is low. Hence, in case that only a few of them are searched and they can be all present in each image, there is not a perceptible processing gain to verify which one of them is present before the search step. However, as the number of entities that are going to be detected (i.e., the total number of known objects in the database) increases, the method becomes more efficient once only a few of them will be present in each scene. The window dimensions are 200x150 (height by width). Figure 3 exemplifies the sliding window process is exemplified. In this figure, the green boxes indicate where the object was recognized. The red box is the final estimated position. It groups the individual parts of the green boxes.

Figure 3 – Example of object movement estimation using the sliding window.



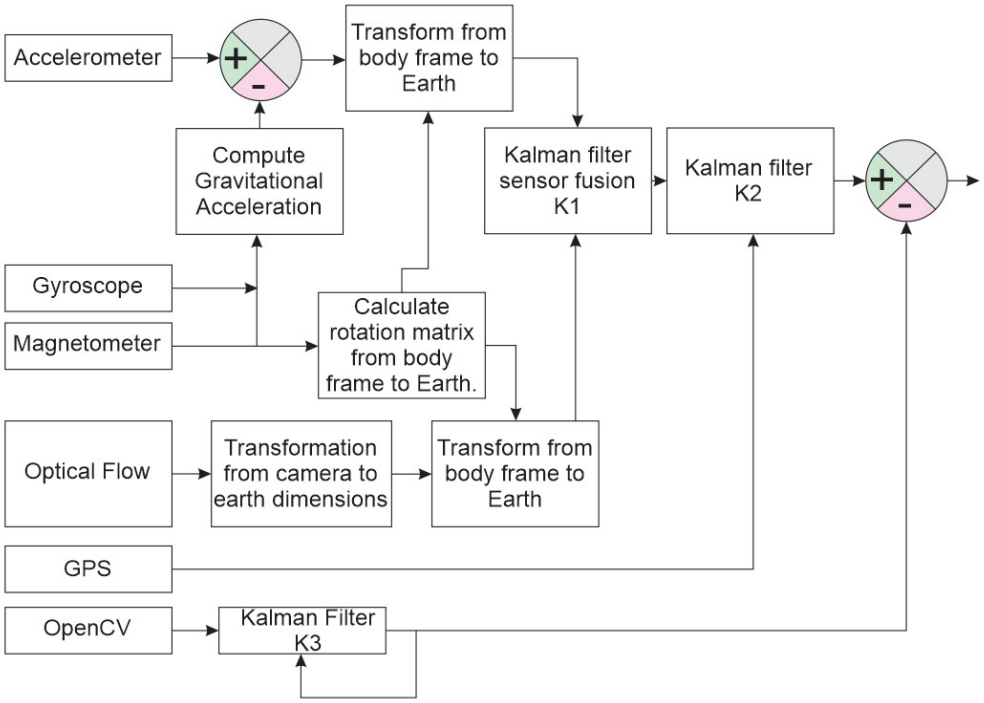
This complexity reduction can be explained using a few variables. N_w gives the number of sliced windows over a defined image, the number of known entities in the database to be recognized is N_d and the number of entities in each image is N_o. If the whole database is searched over the sliced windows, the search time is the function $T_S = f(N_w \cdot N_d)$. Otherwise, the search time is $T_S = f(1 + N_w \cdot N_o)$, which means that the search time is a function of the number of entities multiplied by the number of windows plus a search to determine which entities exist in the main image. The relation between these two methods is given by $r = \frac{N_o}{N_d}$, once $N_w \cdot N_d \gg 1$. Note that usually only a few of the known entities are present in each image ($N_d \gg N_o$). Thus, the second function results in lower search time.

Despite the robustness of current image processing algorithms, errors in the detection process may occur. In this sense, the system should maintain the object monitoring even during the occurrence of occlusions and shadowing while rejecting the UAV movement when estimating the object trajectory. This work uses an Extended Kalman Filter (EKF) to fuse the data from optical flow, Global Positioning System (GPS) and Inertial Measurement Unit (IMU) to overcome these problems. The IMU is onboard the Flight Controller Unit (FCU) and provides fast response assisting the vision whenever it fails due to loss of visual features. The optical flow is a function presented in the OpenCV library, and it extracts image features coming from the visual system. Posteriorly, another Kalman filter is used to fuse data from the previous acceleration result with the GPS to obtain a more accurate UAV pose.

The optical flow information is obtained using the Lucas-Kanade method. This one adds no extra processing once the data was previously used to perform the background subtraction. Due to the fast sampling rate and response time, a physical sensor could be added to the UAV architecture to enable integration with FCU. Although, this calculation would still be required inside the background subtraction algorithm because the optical flow information needs to match precisely with the image sequence to perform an accurate background removal.

In this application, the vision system is sampled at approximately 20 Hz and the IMU at 50 Hz. The Lucas-Kanade method assumes a constant flow around the region of the considered pixel. This method applies the least square algorithm to solve the pixels optical flow equations in these regions. The combination of several sensors data with distinct properties can improve the algorithm performance to calculate the UAV pose despite the data noise. The first filter is responsible for clustering the acceleration information coming from the IMU and optical flow. The second fusion regards the output from the previous fusion with the GPS sensor to improve the UAV pose estimation. This output is compared with the object position coming from OpenCV function to discard the UAV movement and to obtain the entity position related to the earth frame. The earth frame is defined as a stationary frame or the one that it is moving at constant speed. Note that GPS can have a sample rate ranging from 10 Hz to lower than 1Hz, depending on the satellite availability. Figure 4 shows the data flowchart of this process. Note that several transformations are required to combine the sensors outputs proper.

Figure 4 – Flowchart of Object Movement Estimation.



Equation 2.4 presents the definition of a model for the system to build the Kalman Filter properly. The model output is related to a transformation of the previous state of the system and an action of a weighted input. The process model relates the state $x \in R^n$ at a previous time $k-1$ with the current state at time k. **A** is called state transition matrix, **B** is the control transition matrix and u_k is the control input.

$$\hat{\boldsymbol{x}}_{\boldsymbol{\bar{k}}} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}} + \boldsymbol{B}\boldsymbol{u}_{\boldsymbol{k}} \tag{2.4}$$

The measurement model allows the output calculation given a measurement matrix **H** and a noise $\boldsymbol{v_k}$, as shown in 2.5.

$$\boldsymbol{z_k} = \boldsymbol{H}\boldsymbol{x_k} + \boldsymbol{v_k} \tag{2.5}$$

The prediction model $\boldsymbol{P_{\bar{k}}}$ can also be defined in a similar way, i.e., it can be related to the current covariance and the process noise **Q** using the relationship presented in 2.6.

$$\boldsymbol{P_{\bar{k}}} = \boldsymbol{A}\boldsymbol{P_{k-1}}\boldsymbol{A^T} + \boldsymbol{Q} \tag{2.6}$$

The measurement update can be then defined as a relation between the current and last

measurement, as in 2.7, where $\boldsymbol{K_k}$ is a Kalman gain.

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\bar{\boldsymbol{k}}} + \boldsymbol{K}_{\boldsymbol{K}}(\boldsymbol{z}_{\boldsymbol{K}} - \boldsymbol{H}\hat{\boldsymbol{x}}_{\bar{\boldsymbol{k}}})^{-1} \tag{2.7}$$

The gain is given by the average noise of the sensor **R** and the updated process covariance, as presented in 2.8.

$$\boldsymbol{K_k} = \boldsymbol{P}_{\bar{\boldsymbol{k}}}\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{P}_{\bar{\boldsymbol{k}}}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-1} \tag{2.8}$$

At last, the prediction error is going to be computed recursively, as shown in 2.9.

$$\boldsymbol{P_k} = (\boldsymbol{I} - \boldsymbol{K_k}\boldsymbol{H})\boldsymbol{P_k} \tag{2.9}$$

Using the previously equations, it is possible to define the following Kalman filters of the proposed approach, which are k1, k2 and k3 as presented in 2.10. The first filter k1 gives the acceleration fusion. There are two acceleration information in the input of this filter, $\mathbf{H} = [1\,1]^T$. One acceleration is expected as output, meaning that there is no need to define values for the state equation once the data input is the same as the output and there is no physical process or **A=I**.

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}} = \hat{\boldsymbol{x}}_{\boldsymbol{k-1}} \tag{2.10}$$

Once there is no physical transformation in the process, **Q=0**, the **R** matrix is the diagonal matrix formed by the measurement covariance $\boldsymbol{Cov_{Sensor}}$ (i.e., $\boldsymbol{Cov_{OptFlow}}$ and $\boldsymbol{Cov_{IMU}}$), respective related to optical flow and IMU, as shown in 2.11.

$$R = \begin{bmatrix} Cov_{IMU} & 0 \\ 0 & Cov_{OptFlow} \end{bmatrix} \tag{2.11}$$

The k2 and k3 filters are based on the same idea, i.e. to model kinematic equations that estimate the UAV or OpenCV entities filtered position (a_x, a_y, a_z), given its input position, velocity $(\dot{a}_y, \dot{a}_y, \dot{a}_z)$ and acceleration $(\ddot{a}_y, \ddot{a}_y, \ddot{a}_z)$. For k2, the state will be also composed by the angles, allowing the accurate tracking of the UAV orientation. From the idea that the acceleration is not constant, we have 2.12.

$$\boldsymbol{j} = \frac{\boldsymbol{da}}{\boldsymbol{dt}} \tag{2.12}$$

The acceleration itself can be defined relating it to the velocity, as in 2.13

$$\boldsymbol{a} = \frac{\boldsymbol{dV}}{\boldsymbol{dt}} \therefore \boldsymbol{dV} = \boldsymbol{a} \cdot \boldsymbol{dt} \tag{2.13}$$

Replacing for the previous acceleration and applying integration in 2.15, we have 2.14.

$$\boldsymbol{dV = (a_0 + j dt) dt} \tag{2.14}$$

Applying integration from V_0 to V and from 0 to dt, we obtain 2.15.

$$\boldsymbol{v - v_0 = a_0 \cdot dt + \frac{1}{2} j (dt)^2} \tag{2.15}$$

The same concept is valid for the position, where $v = \frac{dx}{dt}$ for the previous found value of the acceleration, which give us 2.16. Then, we can reorganize the equations as in 2.17.

$$\boldsymbol{x - x_0 = v_0 \cdot dt + \frac{1}{2} \cdot a_0 \cdot dt^2 + \frac{1}{6} j \cdot dt^3} \tag{2.16}$$

$$\boldsymbol{x_k} = \begin{bmatrix} x_k \\ v_k \\ a_k \end{bmatrix} = \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \end{bmatrix} \tag{2.17}$$

The state equation will be given by 2.18.

$$\boldsymbol{x_k} = \begin{bmatrix} 1 & 1 \cdot dt & \mathbf{I}_3 \cdot \frac{dt^2}{2} \\ \mathbf{0_3} & 1 & 1 \cdot dt \\ \mathbf{0_3} & \mathbf{0_3} & 1 \end{bmatrix} \cdot \boldsymbol{x_{k-1}} + \begin{bmatrix} \mathbf{M} \cdot \frac{dt^3}{6} \\ \mathbf{M} \cdot \frac{dt^2}{2} \\ \mathbf{M} \cdot dt \end{bmatrix} \cdot \boldsymbol{w_k} \tag{2.18}$$

From this concept, we can extend the interpretation for each axis in the earth reference frame East, North, UP (ENU), such as in REP-105 [76]. The state vector is

$$\boldsymbol{x_{pos}} = [\boldsymbol{x_x, x_y, x_z, \dot{x}_x, \dot{x}_y, \dot{x}_z, \ddot{x}_x, \ddot{x}_y, \ddot{x}_z}] \tag{2.19}$$

Alternatively, when using the angular speed, we have $[\boldsymbol{\phi, r, p, \dot{y}, \dot{r}, \dot{p}, \ddot{y}, \ddot{r}, \ddot{p}}]^T$.The process model can then be re-defined as in 2.20.

$$\boldsymbol{x_k} = \begin{bmatrix} 1 & 1 \cdot dt & \mathbf{I}_3 \cdot \frac{dt^2}{2} \\ \mathbf{0_3} & 1 & 1 \cdot dt \\ \mathbf{0_3} & \mathbf{0_3} & 1 \end{bmatrix} \cdot y_{k-1} + \begin{bmatrix} \mathbf{M} \cdot \frac{dt^3}{6} \\ \mathbf{M} \cdot \frac{dt^2}{2} \\ \mathbf{M} \cdot dt \end{bmatrix} \cdot w_k \tag{2.20}$$

Where $M = \boldsymbol{[(111)]^T}$, $\mathbf{0_3}$ is a 3 x 3 null matrix and $\mathbf{I}$ is an 3 x 3 identity matrix. The covariance matrix $\mathbf{R}$ is formed by the sensor stationary errors represented by the standard deviation σ of each variable. In a practical way, the matrix initial value in the algorithm were adjusted by trial and error. In addition, the $\mathbf{Q}$ matrix will be formed by covariances between

individual acceleration, speed and position variables. In this sense, we have 2.21 and 2.22, respectively.

$$\mathbf{Q} = \begin{bmatrix} \frac{dt^6}{36} & \frac{dt^5}{12} & \frac{dt^4}{6} \\ \frac{dt^5}{12} & \frac{dt^4}{4} & \frac{dt^3}{3} \\ \frac{dt^4}{6} & \frac{dt^3}{2} & dt^2 \end{bmatrix} \tag{2.21}$$

$$\mathbf{R} = \sigma \tag{2.22}$$

Lastly, in this simplified context, the measurement matrix will be then defined dynamically between two possible values for GPS and IMU, i.e., 2.23 and 2.24. The matrix selection is done accordingly to the availability of each sensor data.

$$\boldsymbol{H_{GPS}} = \begin{bmatrix} \mathbf{I}_3 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.23}$$

$$\boldsymbol{H_{IMU}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \mathbf{I}_3 \end{bmatrix} \tag{2.24}$$

The filter k3 deals with more uncertainties than the k2 once the entity recognition may be subject to occlusions and recognition errors. Thus, to overcome the occlusions when the target is not recognized between two scenes, the last predicted information is feedback to the filter. This one is represented as a little dashed line at the bottom of Figure 4. This process repeats into a maximum number of consecutive feedbacks, where the target is declared as missing.

The GPS is based on an earth frame that is different from the coordinate system of the IMU output, i.e., UAV body. So, the accelerometer information from IMU must be transformed to the same GPS coordinate system using a rotation matrix along the ZYX axis. This gives the orientation of the UAV calculated by the FCU of the UAV. Defining a frame f^k, where k is related to the "k" frame reference and the coordinated system (x^k, y^k, z^k), a transformation to another reference frame "n" can be defined as $\boldsymbol{f^n} = (\boldsymbol{R_k^n})\boldsymbol{f^k}$. The variable $\boldsymbol{R_k^n}$ is a transformation matrix that correlates the "k" and "n" reference frames.

The incoming data from the IMUs, GPS and optical flow are transformed into a single reference frame. The input data for the filter follows the ROS Standard REP-103 [77] and REP-105 [76]. This means:

- X – forward direction of body frame points to north;
- Y – direction as left or east;

- Z - as up for body frame and down for earth frame.

The incoming accelerometer information from IMU and optical flow data are related to the UAV body frame [76], which coincides with UAV base movement. The IMU is installed in a place that coincides with the gravity center of the UAV and the camera is as close as possible to it. Hence, all rotation of the platform happens in the center of those sensors. The X, Y and Z axes adopted in the work are shown in Figure 5.

Figure 5 – Reference Axis of the Body Frame.

The matrix transformation that correlates the body frame “b” and the earth frame “E” are defined as $\boldsymbol{I}_b^E$ and expresses the rotation around yaw (Ψ), pitch (Θ) and roll (ϕ) angles of the UAV, respectively. Those rotations are defined in 2.25, 2.26 and 2.27.

$$\boldsymbol{R\Psi} = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

$$\boldsymbol{R\Theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Theta) & -\sin(\Theta) \\ 0 & \sin(\Theta) & \cos(\Theta) \end{bmatrix} \tag{2.26}$$

$$\boldsymbol{R\phi} = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \tag{2.27}$$

2.4.3 SCENE CONSTRUCTION

It is important to create a semantic relation between the incoming data with the process scene understanding. This is required once the information may be disorganized, duplicated

or uncorrelated. A little review of the current literature indicates that a great part of the works in the visual recognition area uses learning techniques with large amounts of spatiotemporal data to learn generic activities [78], [16] and [79]. Note that this amount of information is not always available and may not be desirable. Concerning human activities recognition and understanding, some works are based on Learning by Demonstration remains at the level of signal-to-signal mapping and do not have a genericization capacity [80], [81]. [82] and [83] introduced a grammatical rule for actions. However, these works only involve the manipulation of objects and do not assure the scenes understanding.

As mentioned before, the automata unit produces an output containing a sequence of symbolic information composed by objects and their positions. From this data, a pre-processing stage is necessary before the automata analysis. For a surveillance mission, few actions can be taken from the position of the objects between two scenes, such as "Hold," "Walk," "Run," "Approach" and "Touch," a person. For instance, if the distance between two objects positions is less than a certain threshold (i.e., a few pixels), the system gives the action "Touch". Based on the identification processes and the previously described relationships, the automaton can structure the information.

Initially, during the development of this architecture, the deterministic finite automata (DFA) was used for testing the capabilities of a grammar-based approach to structure the low-level information. However, the DFA has a relevant limitation, i.e., the inability to process the low-level numerical information along with the acceptance states, such as the certainty associated in each recognized instance, which can be an essential factor to the decision maker.

Varied types of automata were defined to allow the development of the quantitative language, such as the weighted and probabilistic ones presented in [18] and [84], respectively. These languages use weights associated with accepted symbols as costs related to the transition execution or probabilities distribution of their occurrences. These approaches limit the application in some real-world issues. Thus, this work proposes:

- Multiple information being propagated through the same automata;
- Automata have associated the recognition certainty of each instance recognized;
- The probabilities can be mapped by weights or normalization of the data.

Few basic concepts based on the weighted automata are defined to aid the development of this modified automata proposal. The automaton associates a weight to every accepted word. However, in the approach of this research work, the transitions are activated by the acceptance of the input symbol along with the verification of the symbol recognition certainty that should be greater than a threshold.

Definition 1. In the proposed approach, $\mathbf{\Lambda}$ is a vector of n-values in such way that $\mathbf{\Lambda} \in \boldsymbol{R} | \mathbf{\Lambda} \geqslant \mathbf{0}$ and it is populated with initial conditions and $\mathbf{M}$ is a binary matrix that

determines if each transition affects the values in Λ. This definition means that the execution of the automata propagates varied information that can be affected by determined transitions or not.

Definition 2. Given a manifold of functions δ defined over a set of states Q and a set of input symbols Σ with a real value associated $|\rho|$. Each input set $\{w, |\rho|\}$ is mapped into a value between $[0, \infty]$.

$$\delta : \{w, |\rho|\} \rightarrow [0, \infty], w \in \Sigma \tag{2.28}$$

Definition 3. The proposed automaton is a tuple:

$$A = \{\Sigma, Q, \Delta, F, q_0, \Lambda, M\} \tag{2.29}$$

With the following accepted language:

$$L(A) = \{w : \Delta(q, w) \in F \,\&\, \delta(\rho)\} \tag{2.30}$$

Where

- Σ is the set of input symbols with associated real values;
- **Q** is a set of states;
- **w** is the input symbol;
- Δ is a table formed by input symbols accepted in determined state transitions with respective manifold function;
- q_0 is a set of initial states;
- **F** is a set of accepted states;
- Λ is a vector of initial values;
- **M** is the influence matrix.

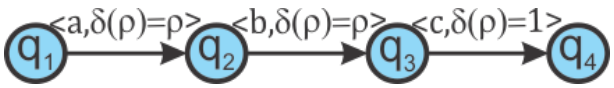
Definition 4. For each symbol accepted in the automata transition ("i","j"), the k_{th} value in Λ is updated accordingly to $M_{(i,j)}$ by a nonnegative binary operation with the output function δ.

$$\Lambda_{new}(k) = \Lambda_{previous}(k) \cdot \delta(\rho), \; if M_{(i,j)} = 1 \tag{2.31}$$

$$\Lambda_{new}(k) = \Lambda_{previous}(k), \; if M_{(i,j)} = 0 \tag{2.32}$$

Example 1. Consider a system that has an event sequence $\Sigma = (a, b, c)$ where each event is associated with a single recognition certainty. The execution must happen in the following order: action $'a'$ with a certainty of 0.8 or $(a, 0.8)$, action $'b'$ with certainty of 0.9 or $(b, 0.9)$ and action $'c'$ with a real value $(c, 5)$, i.e. there is no uncertainty associated with action c. As valid values are associated with the first two actions, a linear equation $\delta(\rho) = \rho$ can be used. In the last action, once there is no uncertainty associated, the output will be always 1 for each input value. This language is accepted by the automata of Figure 6.

Figure 6 – Automata for the Example 1.



An important property of this model shown is the execution equivalence with the DFA. To prove it, consider the DFA as a tuple $\boldsymbol{A_D} = (\boldsymbol{\Sigma_D}, \boldsymbol{Q_D}, \boldsymbol{\delta_D}, \boldsymbol{q_{0D}}, \boldsymbol{F_D})$ and the language accepted by this definition, given symbol input set w^*, as $\boldsymbol{L(A_D) = w^* : \delta_D(q_D, w^*) \in F_D}$. The equivalence of the proposition **A** compared with the automata $\boldsymbol{A_D}$ can be determined if the languages accepted by both are equivalent, i.e., $\boldsymbol{L(A_D) = L(A)}$ [85].

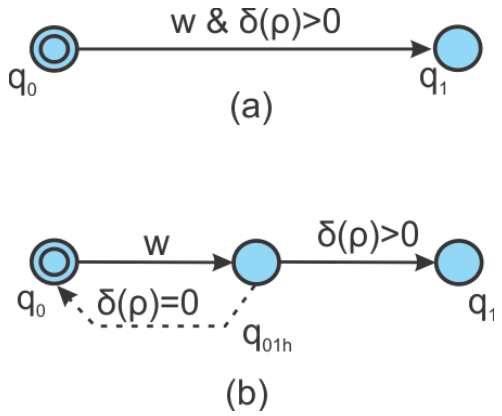
Two states are indistinguishable if for $\boldsymbol{\delta_{D^{\cdot}}(q_{1D}, w^*)}$ and $\boldsymbol{\delta_{D^{\cdot\cdot}}(q_{2D}, w)}$, the following relation is true $\boldsymbol{(q_{1D}, q_{2D}) : q_{1D} \in F \,\&\, q_{2D} \in F}$, i.e., for a set of input, its sequence of transitions results in the same final accepted state.

Theorem 1. If a set of states is indistinguishable, it can be said as equivalent.

Proof 1. The proof comes by contradiction since if there is a way to differentiate the execution between the first and second set of states, it must exist a string $\mathbf{w} \in \boldsymbol{\Sigma}$ that does not satisfy the proposed relation.

The DFA does not directly support the functions used in this new approach. Thus, to adequately demonstrate the execution equivalence between the DFA and the proposed automata, it is necessary to match the symbols. Given the proposed automata defined above, the acceptance state is determined by the associated input symbol 'w' and the function output $\delta(\rho) > 0$, as represented in Figure 7(a). As both pieces of information always arrive at the same time, it is possible to determine which one is verified first as well as to build an equivalent automaton as presented in Figure 7(b), which introduces a hidden state q_{01h}.

Figure 7 – State Acceptance. (a) New Approach Transition. (b) Alternative Representation with Separated Conditions Tests.



At this stage, it is possible to consider the manifold function requirement $\delta(\rho) > 0$ as a symbol w' $\in \Sigma$' which can be defined belonging to the previously defined set of symbols or Δ $\in \Delta$'. As result, if Δ_D=Δ' and Q=Q_D, it must be possible to build a DFA that is equivalent to the proposed model and the DFA properties must be valid due to the fact that every transition in the proposed model can be individually mapped into a DFA.

After the basic definitions, a simplified algorithm, presented in Algorithm 1, is built to show the automaton execution. Note that the proposed structure barely increases the computational complexity of the automata algorithm. This increase in computational cost corresponds only to the activation function calculation and an extra condition in the algorithm required to verify the output function.

Algorithm 1 Automata Execution

Input: Featured automata $A = (\Sigma, Q, \Delta, q_x, F, \Lambda, M)$ with n-staes, and $x = 0$
Input: Input t-strings and values $([e_0, \rho_0], ..., [e_1, \rho_1])$ where $e_0 \in \Sigma$
Output: Final states and values vector q_F, Λ
for $k \leftarrow 1$ *to* t **do**
    for $i \leftarrow$ *to* n **do**
        if $\exists(q_x, q_i, e_k) \in \Delta$ & $\delta_{xik}(\rho_k) > 0$ **then**
            $\Gamma(k) = \Gamma(k) \cdot \delta_{xik}(\rho_k)$
            $q_x = q_i$
        end if
    end for
end for

Example 2. Human activity recognition is one of the most prominent applications for the proposed structure. In this field, a common methodology consists in detecting human poses to determine which activity is recognized. For this work, the example assumes that human activity is determined by observing a set of three individual poses, which do not require a strict sequence. It is possible to recognize the poses by several methods, such as computational vision, accelerometers, among others. However, a limitation in the recognition process arises due to the

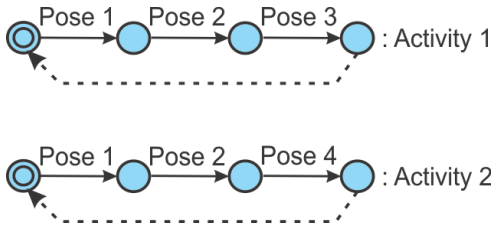
Table 3 – Primary Activities for some Patterns.

Pattern	Primary Activity
Activity 1	Pose 1, Pose 2 and Pose 3
Activity 2	Pose 1, Pose 2 and Pose 4

lack of accuracy in determining the pose, which is explained by the interferences of the sensors that may affect the measurements, resulting in a recognition rate measurement for each detected pose. Table 3 presents the activities.

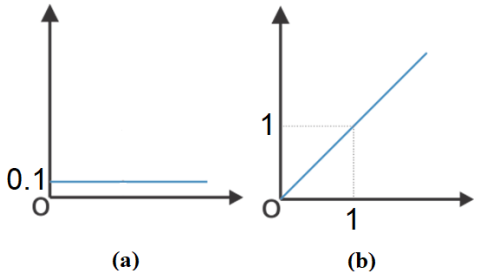
A simple way to represent the proposed scheme is using the standard DFA. Figure 8 shows a sequence of primary activities using a DFA. In this method, the automata do not process certainty. The binary output (i.e., primary activity happened or not) indicates which pose was accepted at the end of the automata execution, and if the algorithm is performed online, it is also not possible to estimate which activity has more recognition certainty before the sequence is completely accepted.

Figure 8 – Activity Recognition Automata.



Example 3. Two activation functions must be defined to determine a model for the new approach. The first function is $\delta(\rho) = 0.1$ (Figure 9(a)). This function corresponds to penalization for a pose that is not part of the activity. The second one is selected as $\delta(\rho) = \rho$ (Figure 9(b)) and it enables direct transfer of the recognition certainty input, allowing its computation.

Figure 9 – Acceptance Function. (a) Function $\delta(\rho) = 0.1$. (b) Function $\delta(\rho) = \rho$.



The structure of Figure 8 can be directly converted to the new proposed approach, being only necessary the addition of the certainty associated in each pose. Figure 10 exhibits the topology for the same problem of activity 1 but more generically. This representation uses a single automaton. Every transition has two associated information: (1) the input symbol; (2) the transition function for every activity. Another difference is the initial certainty recognition.

For the Automata represented in Figure 14, the matrices of $\Delta_{activity2}$ and $\Delta_{activity1}$ are represented in Table 4 and Table 5, respectively. These matrices are essential to the proper execution of the automata once they define which activity certainty will be affected by an input symbol. For instance, in the Activity 2, any input containing pose 3 will be penalized (e.g., the transition from q_{pose1} to q_{pose3} always has 0.1 as output, or, $(\rho) = 0.1$) and for Activity 1, the inputs containing pose 4 will be penalized.

Figure 10 – Proposed Automata for Activity 1 of Example 1.

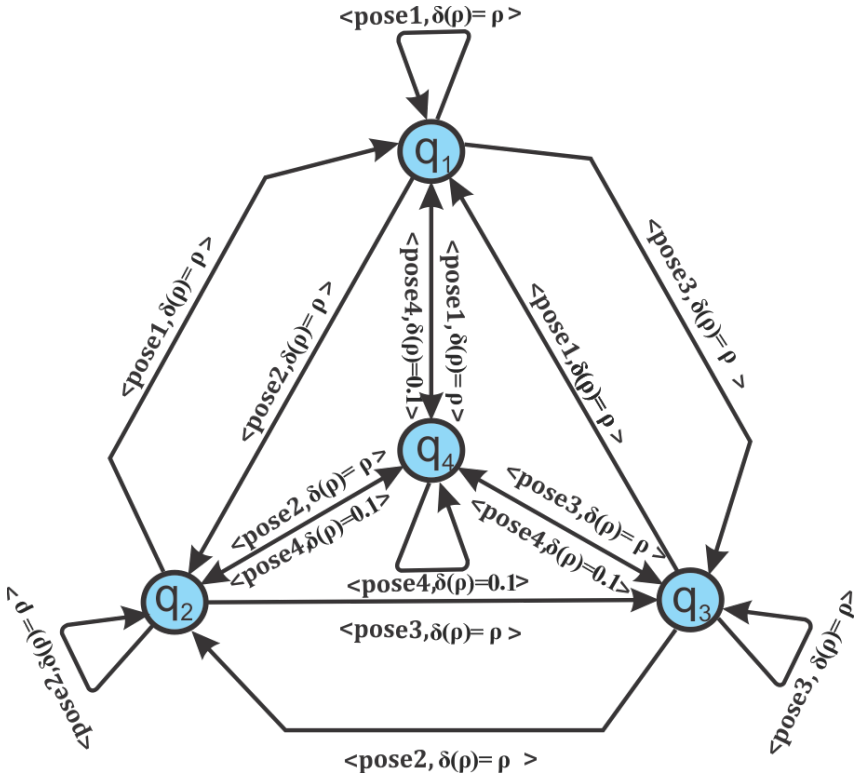


Table 4 – Matrix for $\Delta_{activity2}$.

	q_{pose1}	q_{pose2}	q_{pose3}	q_{pose4}
q_{pose1}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = 0.1)$	$(pose4, \delta(\rho) = \rho)$
q_{pose2}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = 0.1)$	$(pose4, \delta(\rho) = \rho)$
q_{pose3}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = 0.1)$	$(pose4, \delta(\rho) = \rho)$
q_{pose4}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = 0.1)$	$(pose4, \delta(\rho) = \rho)$

Table 5 – Matrix for $\Delta_{activity1}$.

	q_{pose1}	q_{pose2}	q_{pose3}	q_{pose4}
q_{pose1}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = \rho)$	$(pose4, \delta(\rho) = 0.1)$
q_{pose2}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = \rho)$	$(pose4, \delta(\rho) = 0.1)$
q_{pose3}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = \rho)$	$(pose4, \delta(\rho) = 0.1)$
q_{pose4}	$(pose1, \delta(\rho) = \rho)$	$(pose2, \delta(\rho) = \rho)$	$(pose3, \delta(\rho) = \rho)$	$(pose4, \delta(\rho) = 0.1)$

The automaton is executed for three pose sequences to show its operation. Note that each pose has an associated certainty. The sequence is: (pose1,1),(pose2,0.8) and (pose3,0.9). This

execution generates a final recognition certainty for one of the activities (i.e., activities 1 and 2), and the activity that has a higher final certainty is the recognized one, as shown in Table 6. Note that as pose three is not in activity 2, the activation function of Figure 9(a) penalizes the recognition certainty.

Table 6 – Automata Operation for First Input Sequence.

Input Recognition Rate				
Pose 1	**Pose 2**	**Pose 3**	**Final Recognition Rate**	
1	0.8	0.9	0.9	Activity 1
1	0.8	0.1	0.08	Activity 2

In case that the last pose changes, i.e., (pose1,1),(pose2,0.8) and (pose4,0.9), the activity 2 is recognized by its higher probability (i.e., 0.72) as shown in Table 7 as second input sequence. In other example, third input sequence, if an improper sequence containing duplicated poses such as (pose1,1),(pose4,0.8) and (pose4,0.9), the algorithm can still have a valid selection for the output activity, and in the case that Activity 1 does not contain pose4, its related recognition certainty ends up penalized, as shown in Table 8.

Table 7 – Automata Operation for Second Input Sequence.

Input Recognition Rate				
Pose 1	**Pose 2**	**Pose 4**	**Final Recognition Rate**	
1	0.8	0.1	0.08	Activity 1
1	0.8	0.9	0.72	Activity 2

Table 8 – Automata Operation for Third Input Sequence.

Input Recognition Rate				
Pose 1	**Pose 4**	**Pose 4**	**Final Recognition Rate**	
1	0.1	0.1	0.01	Activity 1
1	0.8	0.9	0.72	Activity 2

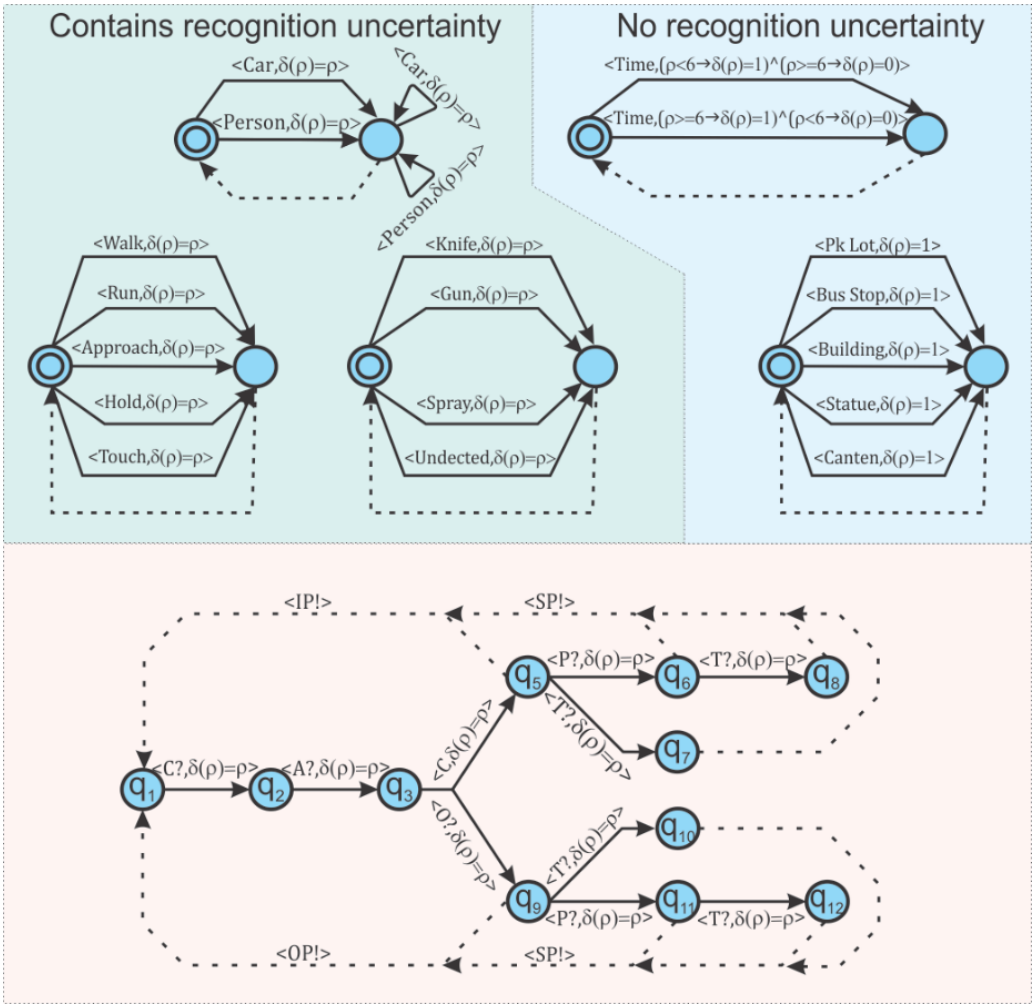
With the definition of the proposed structure, it is possible to introduce a new automata approach model capable of pushing the recognition certainties as continuous value information along with the incoming symbols. For the mentioned surveillance and SAR problem, this implementation is capable of receiving the elements actors (C), actions (A), objects (O), time (T) and location (P). At the end of each sequence, the sequence computes the total recognition certainty. After that, the automaton resets through the dashed line to receive the next data sequence.

Few other definitions must be done to allow the comprehension of the proposal. Firstly, the defined vector for the initial values Λ will start with value 1 and will always be reset to

this value every time that the system receives a new scene. After processing the sequence of transitions, the output value of Λ will contain the overall recognition certainty sequence. Note that as time and place have very a few uncertainties associated with them. Thus, special functions are selected. The data sequence is organized by actors and is always presented as a set containing at least an actor, action and another actor or object, represented as {C, A, {$C \vee O$}}. The scene may also contain place {P} and time {T} as optional modifiers.

The automata operation starts when the first element is feed into the automata (i.e., loops of symbols input). Note that time and place do not have considerable uncertainty associated with them because a little difference in time does not make a relevant impact (i.e., smooth transition) as the difference between "Working hours" and "After hours" (i.e., extreme boundaries). In this case, the transitions are enabled only after or before 6 a.m. (i.e. "Working hours" is set between 6 a.m. to midnight and "After hours" is between midnight and 6 a.m.) to enable the proper transition for "Working hours" and "After hours." The Place has a similar behavior where a little uncertainty between two locations boundaries is not too relevant in the outcome.

Figure 11 – Proposed Modified Automata.



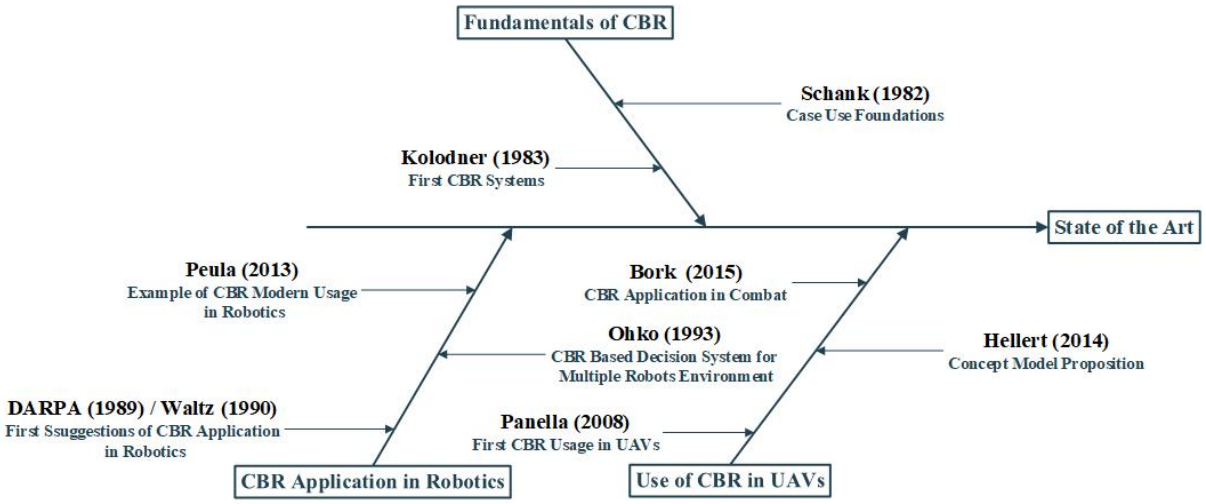
2.4.4 AUTONOMOUS DECISION

The usual methods for autonomous decision are related to systems limited to extensive training over a big set of data. Hence, the current techniques are very far from presenting generic intelligence to operate a broader range of topics [86]. An example is the Google Duplex assistance voice. This voice assistance has an Artificial Intelligence (AI) capable of handling phone calls for scheduling appointments. The person on the conversation may not be able to distinguish the robotic system from a human. However, the Duplex system is incapable to deal with different scopes of conversation in which it was programmed to do.

Critical applications such as aerospace industry require robotics' system ability to comprehend the decision-making process as well as to have predictable and consistent behavior. There are many methods under these requirements. The most used decision-making algorithms are Artificial Neural Networks (ANN), Expert Systems (ES), Evolutionary Computation (EC), Data Mining (DM) and Case-Based Reasoning (CBR) [87]. The DM plays a vital role in decision support systems by modeling and extracting the knowledge accordingly with patterns. On the other hand, the EC deals with more criteria, which is suitable for complex real-world problems. ANN presents the most impressive results. As a drawback, it requires a massive amount of training data. This work uses the CBR method for decision making with the capability of working autonomously in trained tasks. CBR presents advantages such as light computational requirements and good representation capability, which makes it an excellent alternative to the proposed problem.

CBR is a branch of Artificial Intelligence (AI). The decision problems are solved by the re-usage of previous solutions, i.e., cases. This approach was developed by [88] and it lays in foundations on the use of episodes and cases to problem-solving with the first CBR systems dating from the '80s [89]. A simple history of CBR application is in Figure 1. In the robotic field, CBR first appears as a principal component of intelligent systems. This method provides memory and information retrieval for robotic agents in the work of [90]. The work [91] suggested the utilization of CBR in a multi-robot environment, such as for document collection in an office. Several articles follow the evolution of the field into modern applications for storing tasks. For instance, [92] demonstrated the CBR application combined with Learning from Demonstration (LfD) to support a decision in complex multi-robot environments.

Figure 12 – CBR Simplified Field History.

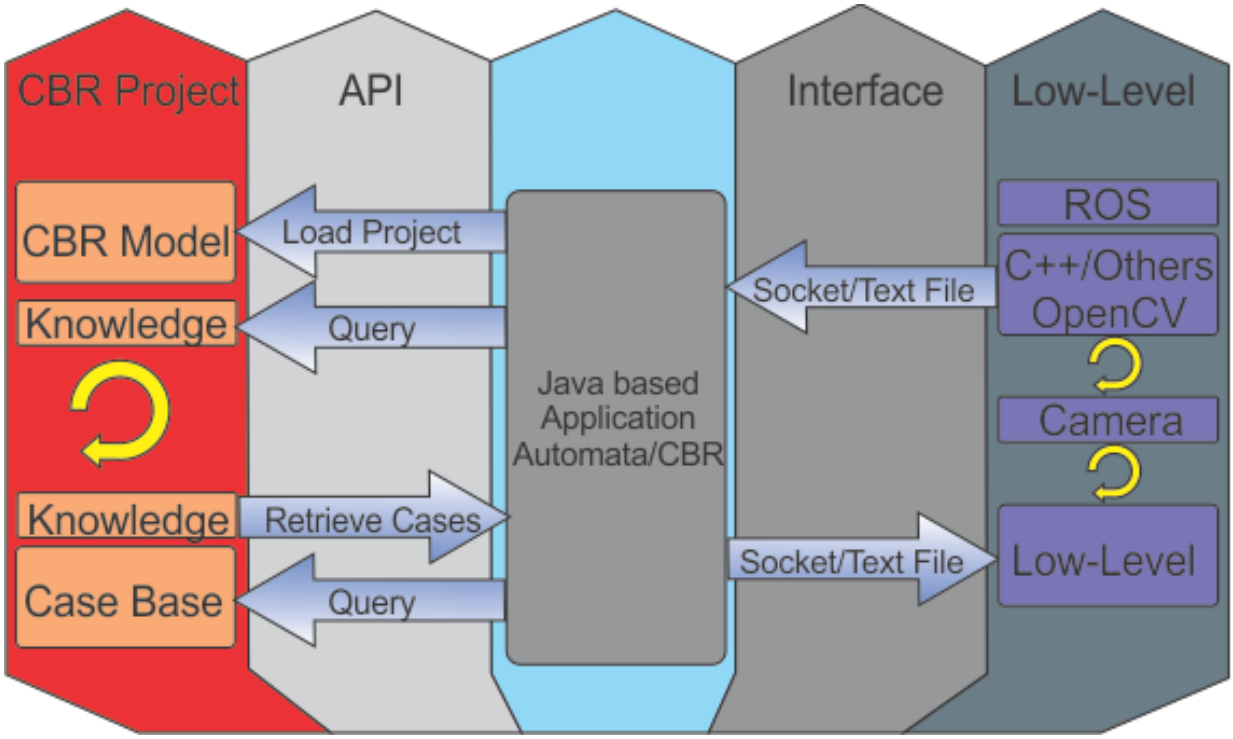


In industry and business fields, CBR has the most comprehensive set of publications and solutions demonstrating modern applications ranging from oil and gas industry [93] to knowledge representation systems [94]. The work [95] is one of the first ones applying CBR in the UAVs scenario. This work analyzed possible CBR applications to have a autonomous system. [96] and [97] used CBR for military and surveillance applications focused on support decision aspects. CBR is chosen partly because it learns through its entire operation [98]-[17] and it takes the natural language output coming from the automata and sensors data to produce a partial view of the UAV world. This cognitive mechanism enables the respective system to perform the activities without human supervision due to scene comprehension.

As the computational vision represents the larger load for the onboard processing and dictates the speed of the processing, the scenes are updated after the image processing algorithms are completed. Each scene is composed by a set of agents present in the frame, potential objects with their respective positions in space, time and by the actions of the agents in relation to previous scenes.

Figure 13 depicts the CBR design . Note that the development of the CBR application takes only a few steps. First, based on the background knowledge and cases collected from the problem, an appropriate case model representation and an initial case database are built. The loop in this figure indicates that the models should change following the new data in the database and new words will require new values of similarity. In this stage, an adequate similarity measure and good retrieval functionality are crucial. The API stage specifies the interaction of the components of the software and is responsible for programming the graphical user interface (GUI) as well. The application in JAVA was build using myCBR [99], which allows an easy combination with other applications and extensions, e.g., similarity calculations and case representation from existing raw data. There is a socket interface to exchange information with the C++ algorithms implemented using ROS libraries [100].

Figure 13 – CBR Design.



Notwithstanding the foregoing, a case representation describes a given scene of the UAV world. The cases are triplets formed by "Scene Information," "Past decisions" and "Expected behavior." The scene information contains agents, objects, perceived actions and real/discrete values that represent continuous variables. The past decisions operate as a kind of memory, representing previous solutions to allow the UAV to provide a smooth response in a very dynamic environment. At last, the expected behavior shows the most suitable solutions for the case. There is a specified conclusion in the record system that will be used by the UAV to determine its action for each scene in the database. Few possible outcomes are available to the UAV decision system:

- **Inform:** HMI will alert the human operator for a specify threat;
- **Inspect:** the UAV should set a flight plan to increase the certainty of given object recognition as well as to visualize a possible outcome;
- **Register:** the scene may contain a potential problem, but does not require the immediate attention of the human operator;
- **Track:** the UAV should prioritize tracking a determined object;
- **Request Backup:** this action should be taken if the UAV perceives that may not be possible to complete the mission;
- **None:** no action is required for the given scene.

The intention is to inform the user as close as real-time as possible about all the UAVs. The user may intervene in the decisions at any moment and in case of noticing that the decisions made by UAV are not proper in accordance with safety proceedings. This process is the main learning mechanism of the CBR, and at this stage, there is no adaptation mechanism implemented. Once the user starts the intervention, the database creates new cases containing the outcome selected by the user, and an equal case will be removed if it already exists.

The similarity of the incoming cases needs to be defined. Hence, there is a necessity of measuring how the incoming cases are compared against the database. The correctness of incoming information is an important assumption, once the semi-autonomous system fully generates the information presented in Figure 48. It is assumed that this information does not contain any grammatical error like a misspelling. This simplifies the assumptions necessary when building the similarity function. In this research, the similarities for the CAD and decision making of Cognitive Level 1 are calculated according to 2.33. The parameter $\phi(s_d, s_c)$ is the similarity between a symbolic value in the test case s_d and the reference case in the database s_c. This parameter is responsible for retrieving the information in a determined table. The variable r_c is a real value in the test scene and r_d is the value in the reference case. The similarity between them is measured by the quotient $\frac{r_c}{r_d}$. Considering sim(c,d) as a similarity between the test case and the database, the weight of an object type in the CRB field is represented by $\omega_0 \subset (0, 1)$, and the normalized value of the similarity $sim_N(c, d)$ is assumed as in 2.33. This similarity measurement was chosen due to the availability in the application library used to implement the architecture.

$$sim_N(c, d) = \frac{\sum(\phi(s_d, s_c)) \cdot \omega_0 + \sum(\frac{r_c \cdot \omega_0}{r_d})}{max(sim(c, d))} \tag{2.33}$$

The computational cost of the proposed model is not relevant once its complexity increases linearly with the number of parameters that are added. The CBR computational cost is also much lower than image processing stages. Once the similarity is calculated for the whole database, the two most similar cases are recovered, i.e. $sim_{N1}(c, d)$ and $sim_{N2}(c, d)$. After that, if the difference in similarity between the recovered cases is lower than a reference threshold T_s, i.e. $sim_{N1}(c, d) - sim_{N2}(c, d) < T_s$, the outcome with lower implications in the current UAV plan will be chosen.

2.5 PARTIAL CONCLUSIONS

This chapter showed an overview of cognitive architectures for a wide range of applications. It also highlighted the advantages and disadvantages of the most cited related aerial architectures. An overview and discussion of the algorithms used for the proposed ARCog was also presented.

The chapter presents a brief review detailing the advantages and constraints of fog computing. The fog-cloud framework was inserted in the ARCog to optimize three subproblems when applying a fleet of UAVs for a determined task: latency, throughput and power consumption. It also compared the characteristics and models of these subproblems concerning the literature.

A methodology using the well-known EKF to predict the trajectory of a moving target and the UAV movement without human intervention is presented. Firstly, the EKF fuses the data from the optical flow with the information of the IMU, which is onboard the FCU. Posteriorly, another EKF combines these accelerations with the data from GPS to result in a more accurate pose for the aircraft. At the end of this process, the output is compared with the object position obtained through an OpenCV function, and the system discards the UAV movement.

An innovative modified version of the weighted automata is introduced. This algorithm enables the propagation of many information through the same automata, and this modified approach associates the certainty of each recognized instance, and the probabilities can be mapped by weights or normalization of the data.

Due to the many challenges and opportunities in these areas, in the last decades, several works were published concerning these research fields. Figures 14 and 15 show the number of researches including the keywords “Cognitive Architecture” and “Cognitive Architecture and Robotics” in the IEEE and Elsevier databases. Note the growth in the number of articles published in conferences, journals, and books over the years.

Figure 14 – Number of researches in the IEEE and Elsevier databases including the keyword: “Cognitive Architecture”.

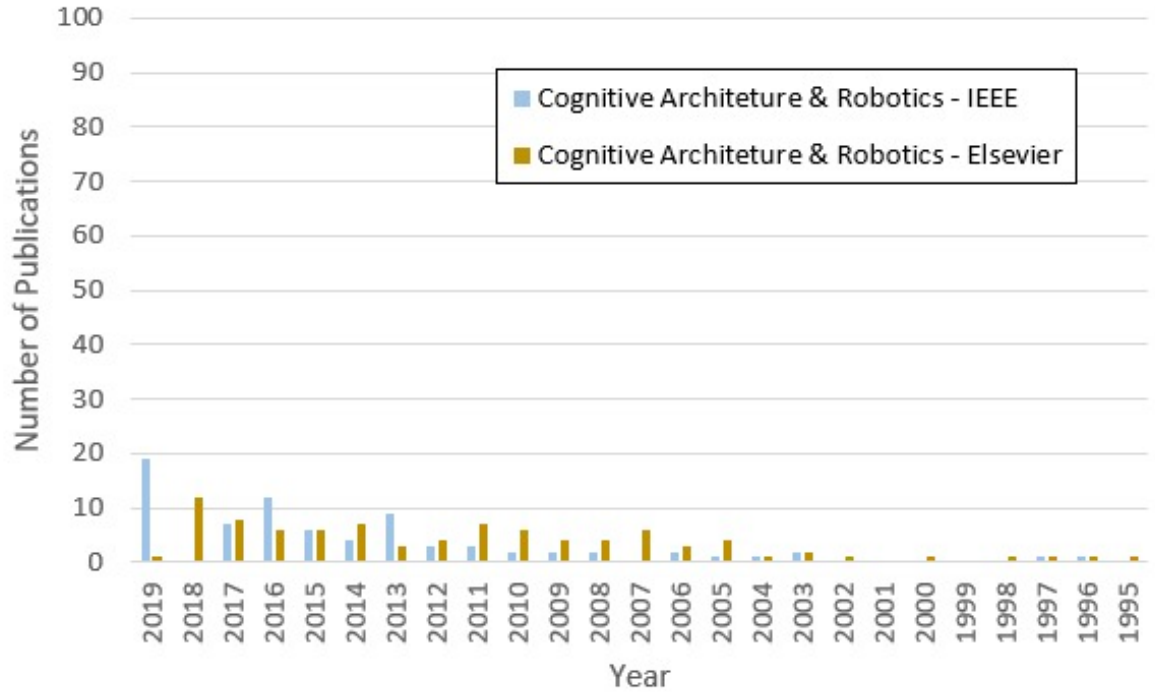
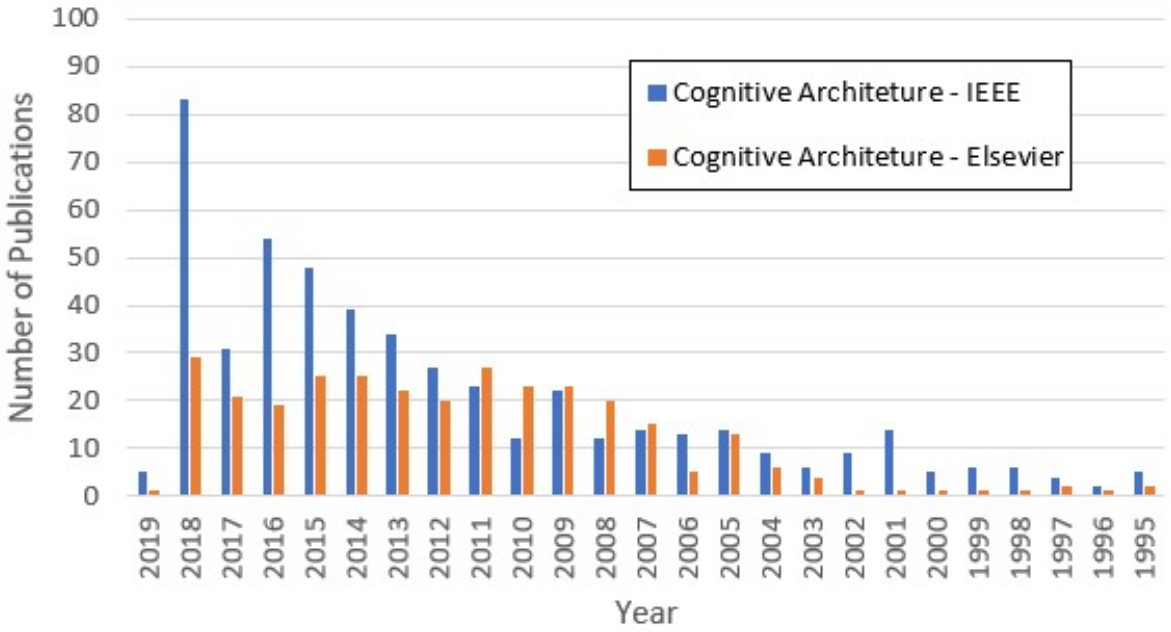


Figure 15 – Number of researches in the IEEE and Elsevier databases including the keywords: “Cognitive Architecture” and “Cognitive Architecture and Robotics.”



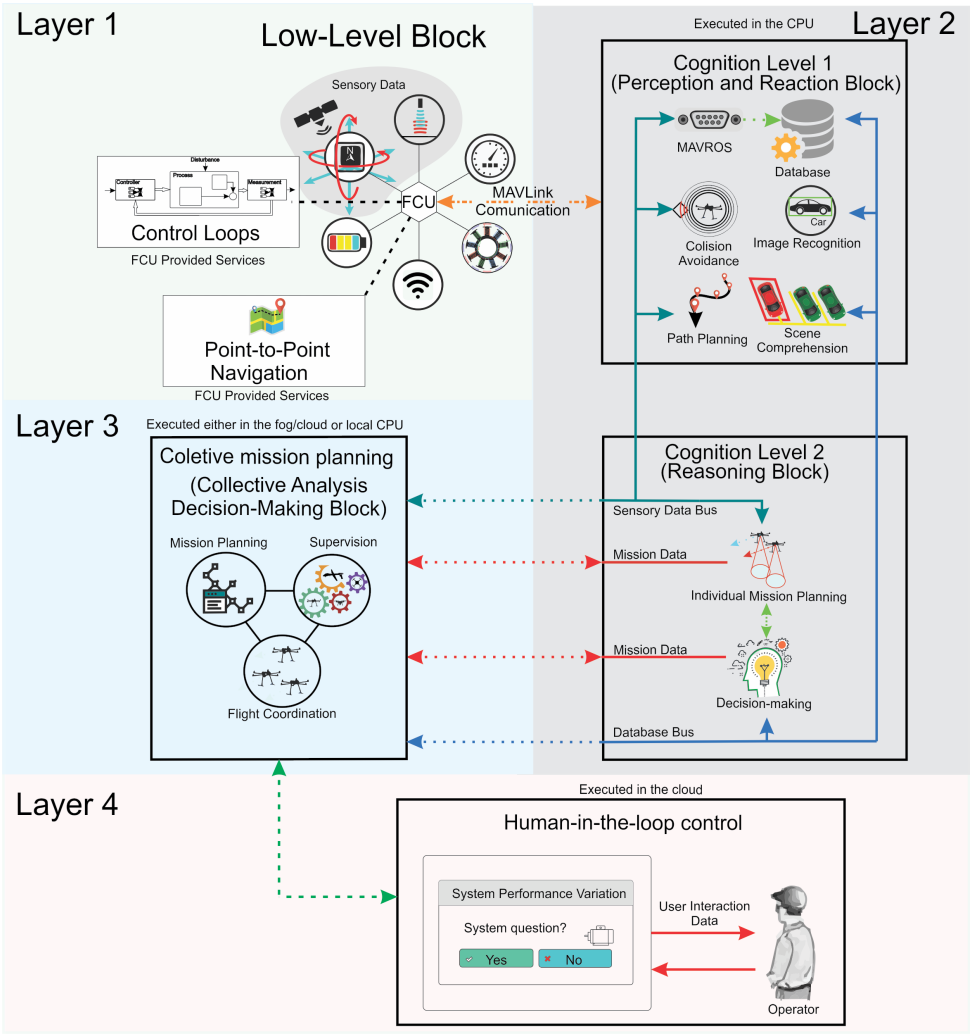
3 AERIAL ROBOTICS COGNITIVE ARCHITECTURE

This chapter describes the innovative multi-level cognitive architecture for UAVs, coined as ARCog. The main objective is to build an architecture oriented by goals and tasks that should be capable of easy adaptation in different scenarios and situations. The architecture should be capable of learning about the operational environment instead of being pre-programmed every time a new situation arises.

3.1 ARCOG

The ARCog organization is a central point for achieving the objectives of improving understanding and boosting performance. The proposed multi-level organization is centered on the cognitive level paradigm [101]. Figure 16 exhibits the logic diagram. The data flows vertically among layers while the inter-layer communication occurs in a hybrid fashion, where some data can also flow horizontally.

Figure 16 – Logic Diagram of the ARCog.

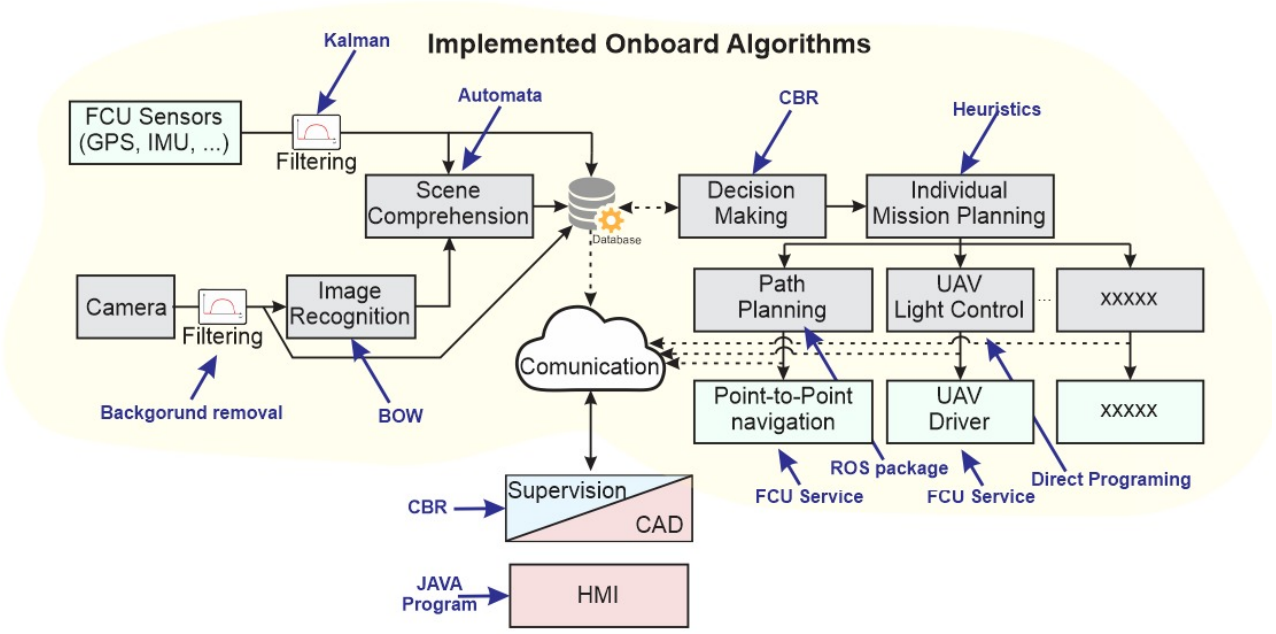


The Low-Level Block, presented in layer 1, is composed of the hardware interface (e.g., IMU, GPS, Ultrasonic sensor) and primary control loops to keep the UAV guided with a stable flight. Besides, it also contains data acquisition to the next levels. This level should be seen as a hardware abstraction layer. Note that there is only one hard requirement, i.e., the existence of a proper interface for the Cognitive Level 1 on Layer 2. The second block (i.e., Cognitive Level 1) performs image recognition, data storage, scene comprehension and basic heuristics (e.g., path planning). The autonomous decision making and individual mission planning should appear at Cognitive Level 2, which uses the information generated at the previous layers for high-level decision making. At the end of this process, the Collective mission planning should be capable of collective analysis decision making among all aircraft, strategic plans, comprehension of the system goals and performing of social skills, i.e., dialogue between UAV and the human operator. This interaction is given by the HITL inserted through a human-machine interface (HMI). The architecture can send data to an external operator, presented in layer 4, for additional processing. Moreover, the ARCog can receive the necessary information through the human agent for changing its mission or the analysis of objects that do not have an appropriate recognition.

The architecture runs multiple processes in a multitasking operating system through ROS (acronymic for Robotic Operating System) communication methods. The ROS was chosen as a meta-operating system since it is an open-source platform and has a collection of software libraries and tools. This operating system assists in robot application development and promotes code re-utilization. The mechanisms provided by ROS are the publish-subscribe schemes and request-replay services. The multi-task operation is important to execute higher and lower processes with different frequencies — for example, the path planning algorithm demands execution as close as real-time as possible and more computational cost.

Figures 17 exemplifies the architecture interactions. This diagram details the sensory information processing. First, the FCU publishes sensory data from its flight control structure, such as the UAV attitude information (e.g., position and orientation). Parallel, the camera connected to the CPU publishes its images to the ROS structure. All this data is stored in a flight survey database for recovery, processing or for informing the ground station. For instance, the decision making recovers the newest scene available and the previous ones from the database. Then, the CBR (i.e., inside this unit) calculates the similarities from the cases database and determines the best outcome. The Individual Mission Planning receives the decisions and enforces the units responsible for those actions. For example if the decision making outcome generates a new path planning, the Individual Mission Planning has to take points or area where a path is required. Then, it should call the Path Planning unit that will make the required action and pass the info for the appropriated low-level interface or control algorithm. In this case, the path will be feed into the FCU point-to-point navigation to interface with the motors control loops.

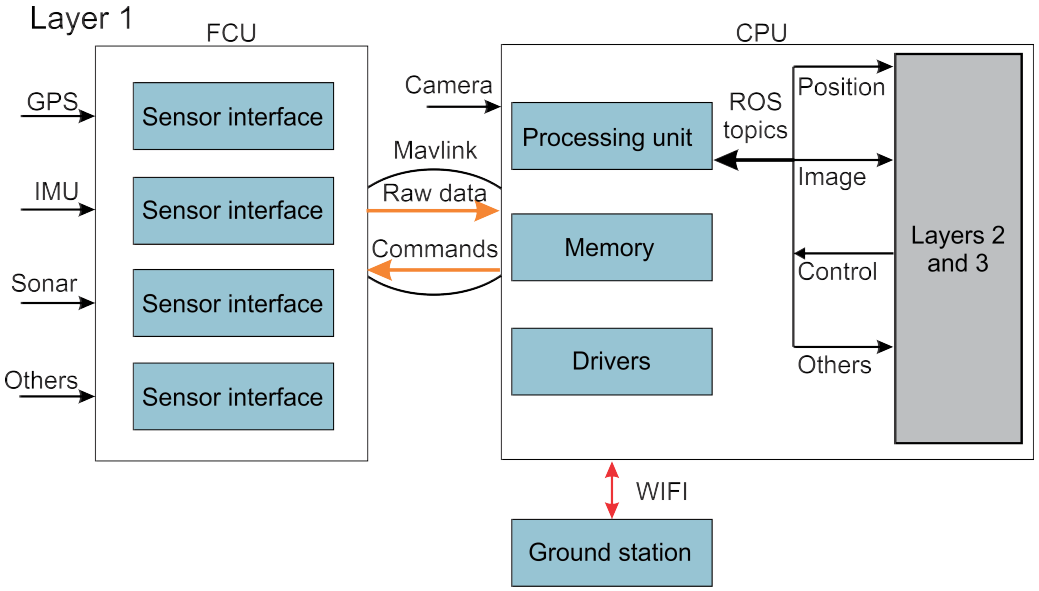
Figure 17 – Flow diagram of the ARCog processing sensory information.



3.1.1 LAYER 1 (LOW-LEVEL BLOCK)

This layer is an abstraction of the hardware. It is equipped with basic features to acquire the required data to feed the other layers. This level should allow control based on high-level decisions without human intervention. Note that most of the current UAVs generation already packs powerful microcontrollers capable of reading different sensors. Moreover, they can control flight parameters to maintain a stable flight and to perform point-to-point control from USB or serial interfaces. These features are particularly useful once they abstract the hardware interface and they can enable the focus on the cognitive aspects of the system. Figure 18 details the model organization.

Figure 18 – Low-Level Block and the relationship with the rest of the ARCog components.



The Flight Control Unit (FCU) is one of the central components. The inputs of the block are sensor data, and the FCU contains all described basic features. Other resources may be required by specific applications and should be provisioned accordingly. For example, another essential component is the Control Processing Unit (CPU). The CPU gathers the data from additional sensors (e.g., camera) required by the upper cognitive levels. This should have a required power processing to execute the algorithms and enough power to allow the onboard application, the CPU, to exchange data with the FCU through the MAVlink protocol. The data exchange utilizes the ROS interfaces. This interface is composed of named information buses called topics.

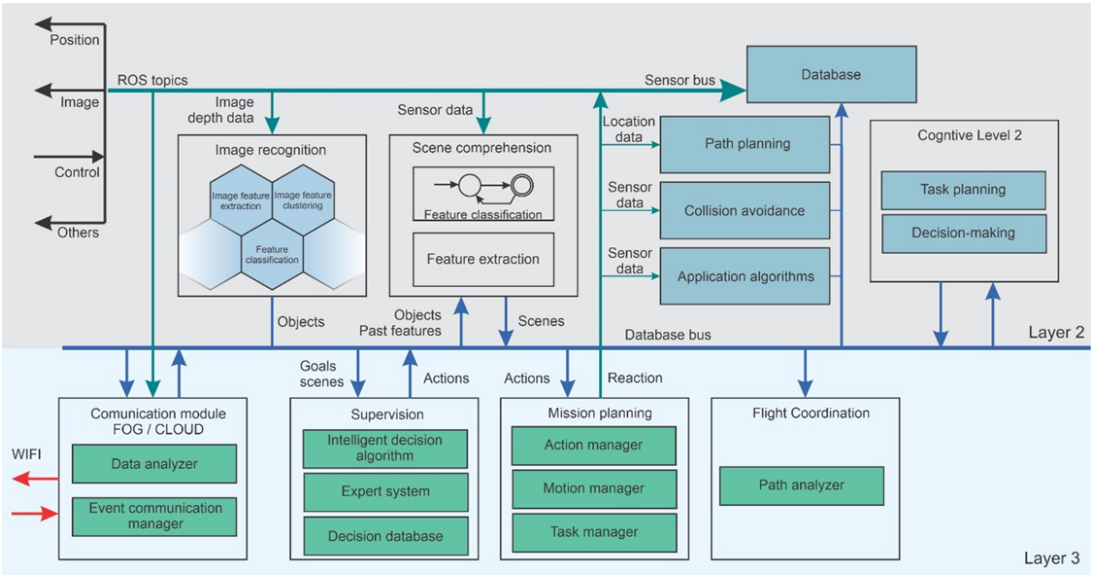
The CPU is also responsible for the interface between aircrafts (Air to Air - A2A) and between aircraft and the ground station (Air to Ground – A2G). The CPU captures data from sensors that are not supported by the FCU (e.g., images). Besides, it executes all codes related to the higher levels and provides storage required by the architecture.

3.1.2 LAYERS 2 AND 3 (COGNITION LEVELS)

The previous layer has detailed the necessary services at the FCU, sensory inputs, and the interface between the abstraction of the hardware layer and the cognitive structures. The cognitive structure of layers 2 and 3 has been designed based on Bloom's Taxonomy of Learning Domains to produce a hierarchical structure related to the task generation, scheduling, execution and supervision [102]. This process results in a formation centered on the cognitive process. Bloom's Taxonomy has been successfully applied in many educational fields to structure courses, learning goals and activities [103]. This architecture provides an insight into the building of human knowledge. For this reason, the taxonomy can also be used to organize the architecture's blocks, where the knowledge is created and improved from the input sensory data up to the

complex thinking of goals and decisions. The cognition is strictly linked with the capability of perception, reasoning, and action. The proposed structure sits right over the sensory-motor blocks of the UAV (Low-level block). The architecture is organized in such a way that, at each layer, the data is processed to produce information and knowledge at the increasing level of complexity. Figure 19 details the cognitive levels and the relationship with the rest of the proposed ARCog components. The inputs of the cognitive levels come from sensor data published by the MAVros.

Figure 19 – Generic description of the cognitive levels and their relationships with the ARCog.



These layers have the desired performance, behaviors, and goals that should be used to generate tasks. Layer 1 contains methods that are more related to data processing, such as, path planning, object recognition, and scene comprehension. Other applications may require different features. For example, marker-based localization and servo control could be appropriately accommodated at this Layer. This block also presents an image recognition algorithm to extract information from the environment. Other methods represented in this figure are individual mission planning and collision avoidance, which were positioned to symbolize common algorithms related to the system operation. An important concept, applied in this work, is the scene. A scene is a dataset containing the content present in the UAV camera visual field, such as objects and actors and their positions, along with all other available sensory input. This is a loose definition, and other works may make use of different features

The scene comprehension in Figure 19 extracts essential relationships from the scene data, analyzing the information from sensors and outputting the relationships between them. For example, the system can obtain the changes in the position between object and person to determine if a given person is holding an object or if there is an object exchange. Several methods can be applied to fulfill this function. Despite the efficiency and easy programming, the non-symbolic method neural network is not appropriate for the specific application of this work. This issue is previously discussed in Chapter 2. Non-symbolic methods do not allow a proper

evaluation of the logic used to extract important data. Thus, grammar-based methods have a substantial advantage with respect to other propositions once their outputs are correlated to the human experience. This research uses an automaton for building scene comprehension.

This research uses the modified automata presented in the previous Chapter for building the scene comprehension. The main advantage of using this kind of method for aerial application is the explicit organization of the pieces of information, and it can remove the redundancies to extract the relationship for high-level analysis. This process is highly depended on the context and tasks once the relations among data changes in accordance with those parameters. Note that few other methods could be applied with some degree of equivalency. An example is the Petri networks. However, automata have a strong representation capability.

The Cognition Level 2, i.e., Reasoning Block, takes the information from the previous block to produce a partial view of the UAV world. This incomplete scene is compared against a database of cases for decision making and task planning. The system builds the case model based on the background knowledge and the previously collected cases. A detailed explanation is reported by the authors in [104]. This process uses CBR method. This approach uses specific knowledge of real situations previously experienced. Those can be adapted to new situations. The CBR is applied based on the built scene through sensory data, image recognition, automata organization and by the last taken decisions, i.e., this layer executes the CBR algorithm each time that the database saves a new completed scene.

The first cases used for decision making are fed into the system in a learning phase. However, they can be online adjusted through the inputs of the supervision system. In this architecture, local decision making does learn or outputs direct actions. Instead, the decision system learns the broader decisions, and the resultant chain of operations have to be further evaluated. This means that CBR's decisions are later processed by task planners to transform these pieces of information into proper inputs to algorithms responsible for its execution. For example, if the UAV decides to follow an object, then, the motion planning must determine the object position to start the path planning for fulfilling the new decision. This process simplifies the implementation of the CBR.

The Cognitive Level 3, i.e., Collective Cognition Decision Making Block, is responsible for the generation of tasks made by the mission planning, team organization and individual tasks attribution performed by the flight coordination unit. Moreover, this layer monitors the tasks' execution and interacts with the HITL for supervision. This layer will be usually encoded in a centralized system or a fog computing architecture. As the architecture can be used to an individual or multiple UAVs in a hybrid way, the fog structure has the implementation for supporting features for cooperative work. The mission planning has to ensure a proper share of mission goals among the available aircraft. Flight coordination has to assure that different UAVs path do not overlap with each other in any location to guarantee flight safety.

The autonomous decision making enables the UAV to respond to environmental changes

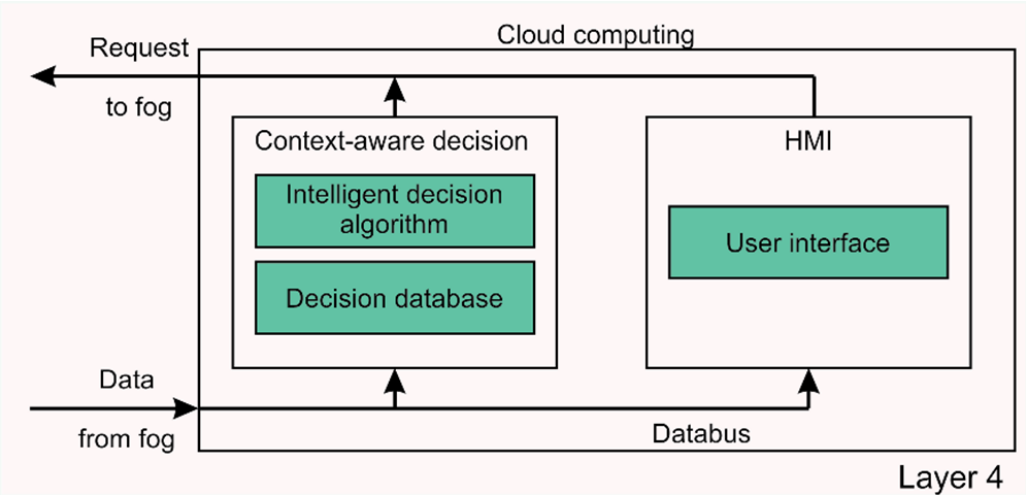
and tasks requirement during the mission execution. In Cognitive Level 3 block, the task planning determines the flight parameters for executing the tasks and other related activities performed by the aircraft during its flight. The decision making should take decisions related to the path planning, interventions, and actions in respect to environmental changes. Besides, it should also be capable of improving the decisions during the flight.

Note that these proposed layers can be implemented with different algorithms and methods. The suggestions shown here were mostly chosen by their relation between computational cost and performance. Hence, the capability to embed the proposed structure depends on the selected methods. The following subsections are going to detail some of these algorithms.

3.1.3 LAYER 4 (HUMAN-IN-THE-LOOP CONTROL)

Activities with long term data storage and human supervision are the responsibility of the last layer in the cloud. The human operator accesses the information by a human-machine interface (HMI) unit, and the other one, the context-aware decision making (CAD) unit adapts the various aircraft information to ease the operator's action. Every time the operator intervenes, the HMI stores that particular action along with the data presented in the unit. Hence, the next time the same set of aircraft data is received, the architecture can prompt the operator with actions to be taken or even start them automatically. In the case where the UAV is not responding for a given amount of time, the CAD can decide to alert the operator or take another action based on a database of decisions, such as action plans. In the same example, the CAD could help the operator by presenting an action plan for this particular task, e.g., sending a command to the unresponsive UAV to return to the initial place [105]. Figure 47 details the HITL layer.

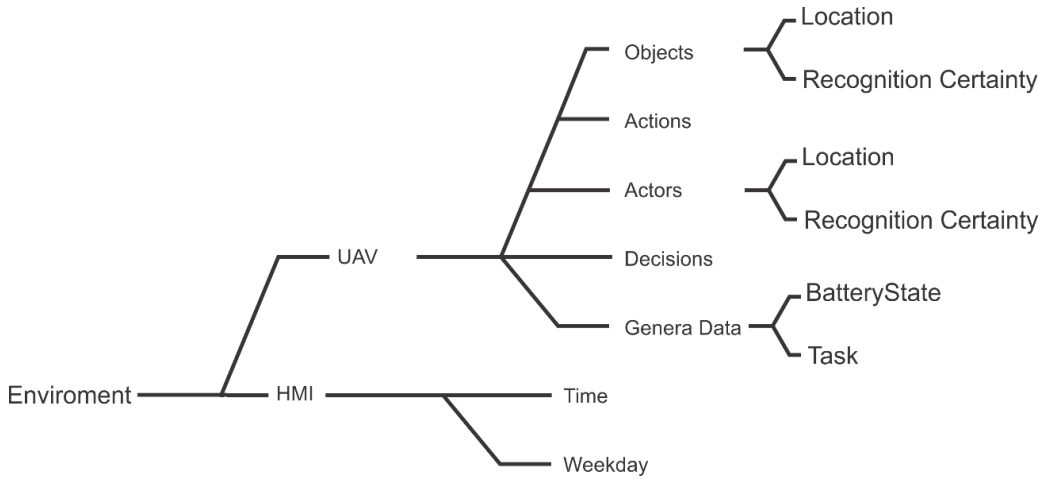
Figure 20 – Human-in-the-loop scheme.



The CBR of the CAD unit deals with the data from several UAVs (e.g., sensory data, detected actors, objects and actions) along with the operator past decisions. Figure 48 shows the

data hierarchy. With this information and a database of cases, this unit will assist the operator in case of a similar situation.

Figure 21 – CAD Data hierarchy.



3.2 PARTIAL CONCLUSIONS

This chapter has described a hierarchical cognitive architecture for UAVs called ARCog. The architecture lays the groundwork for the development of a software platform aligned with the current requirements of state-of-the-art technologies. ARCog is designed to deal with high-level decision making, such as understanding, interpretation and goal-oriented reasoning.

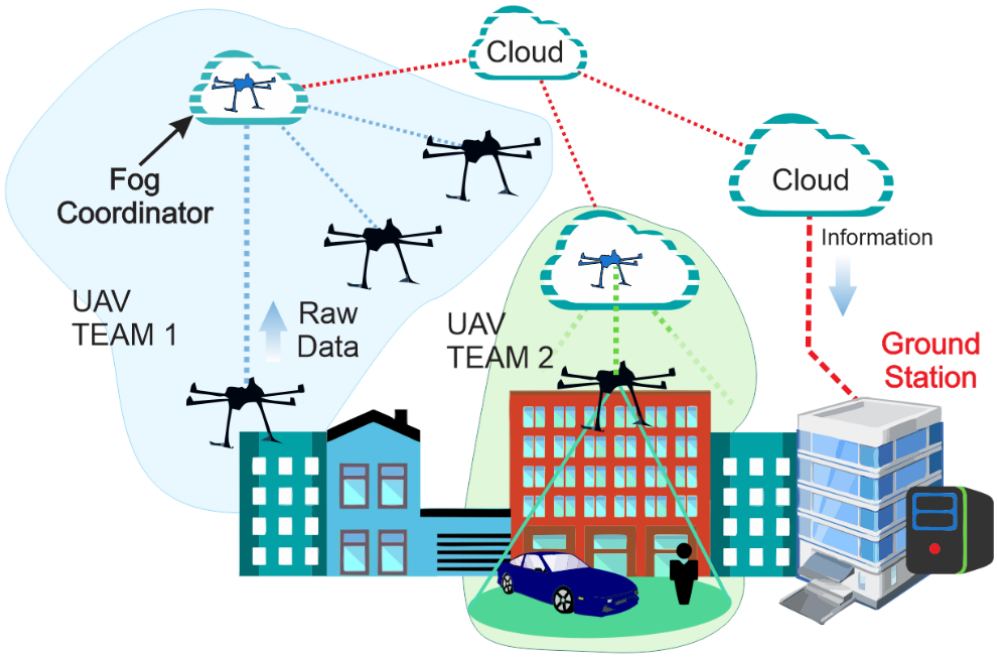
ARCog is based on Bloom's Taxonomy of Learning Domains. The layers can be hierarchically arranged from low-level perception to high-level cognition by a modular design to tackle problems related to specific blocks. Processing data produces information and knowledge at increasing levels of complexity. The several layers in the architecture are capable of abstracting hardware, and they encapsulate the data flow from perception to reasoning.

The architecture formalization includes five main blocks distributed in four layers. These components guarantee the execution of the architecture for semi-autonomous missions. The modules were developed to solve specified tasks. They encapsulate a certain level of abstraction.

4 FOG-CLOUD COMPUTING FRAMEWORK

Figure 22 illustrates the problem representation of a fog-cloud computing collaboration. Each UAV is responsible for determining its trajectory autonomously based on the assigned task. Besides, each UAV captures images and information required for its missions. During the task execution, some commands are the responsibility of the GS, for example, supervision, position monitoring, data video storage, and task assignment. These basic definitions lay the groundwork for defining the architectural problem, i.e., defining how components are organized. The next topics discuss the main aspects related to the proposed problem.

Figure 22 – Problem representation for the fog-cloud framework.



4.1 FOG NODE LOCALIZATION

The localization of the fog computing node is a key component in the architecture definition. There are two considered possibilities. First one is the placement of the fog node on the ground, geographically near the UAV group. The second possibility is to embed the fog device in the UAV coordinator that will move together with its team during the mission execution. The main benefit of the first option is to avoid power restriction due to the possibility of connecting the fog device to power stations or generators. As a tradeoff, this configuration creates data transmission restrictions once the objects on the ground can interfere with the network signal.

The deployment of a fog node makes the UAV architecture flexible and more autonomous. An example is the flight time of the head coordinator that does not affect the mission due to the fact that the all aircraft executing the missions share the same power restrictions.

4.2 FOG NODE SERVICES

Initially, the first implementations of fog computing were responsible only for information clustering. However, the current applications pack several types of preprogrammed services capable of handling the incoming information directly at the fog device. This work uses the second approach. The head coordinator will manage the information of its group to provide services and to cluster the information into the cloud.

It is important to know the requirements of UAVs to determine which services are necessary at the fog level. In this work, the aircraft can operate autonomously; i.e., they can perform flight control and data gathering and take decisions regarding their tasks. Especially for SAR context, the UAVs need to flight along a certain path to capture images for objects recognition. Besides, they need to decide whether the mission is accomplished or not based on the acquired information. An architecture capable of supporting this level of automation was proposed by the authors in [104].

In many architectures, the GS assists the aircraft when activities require a certain level of high cognition. Usually, the supervisory system performs task planning, monitoring, path planning, among others. However, most of these activities do not require direct human intervention and can also be performed by another autonomous system. In the approach of this work, some services are deployed on the fog computing node whenever is possible to reduce the fog-cloud throughput (e.g., data storage). Other tasks such as the mission goals definition and supervision are processed in the cloud.

The proposed task distribution requires that important information is shared with the cloud and a great part of the data stays at the fog level. In this sense, an algorithm must classify the incoming data to determine which ones should be sent to the cloud. Based on the premises that each UAV can take decisions related to the task execution, it is possible to say that critical mission data can also be flag accordingly to its importance. Figure 22 represents this explanation. A coordinator located at the fog level manages the information of the nearest nodes. The data that is not processed and stored locally in the coordinator is forwarded to the supervisory system located at the cloud for further processing.

4.3 FOG NODE CHARACTERISTICS

The hardware characteristics are important to conduct the analysis of the proposed architecture. Initially, the methods for data transmission regarding the inter-UAV and fog-cloud communication are selected. Each technology should provide different data rates and transmission ranges.

Table 9 shows a list of key characteristics for the typical wireless communication technologies. The selection of proper hardware for UAV application should consider the relation between energy/coverage for the large flight times.

Table 9 – Characteristics of typical wireless standards [56], [106], [58], [107]

	Bluetooth	**Wi-Fi**	**HSPA**	**ZigBee**
Coverage	100 m	0.1 – 2 Km	5Km	1.2 – 14 Km
Throughput	22 Mbps	Up to 300 Mbps	5.76 – 11 Mbps	0.25 – 72 Mbps
Frequency	LF 120-134Khz HF 13.56MHz UHF 850-960MHz	2.4, 5GHz	TDD 1.85-3.8GHz FDD 0.7 – 2.6GHz	0.9, 1.2, 2.4 GHz
Energy Efficiency	High	Low	Depends on the signal strength	Depends on the model

Table 10 – Common characteristics of SoC [108], [109].

	ARM	**x86 / x64 + GPU**	**FPGA**
Task Efficiency	Medium	High	Low
OpenCV Efficiency	Low	High	Medium
Energy Efficiency	Medium	Low	High
Implementation Complexity	Low	Low	High

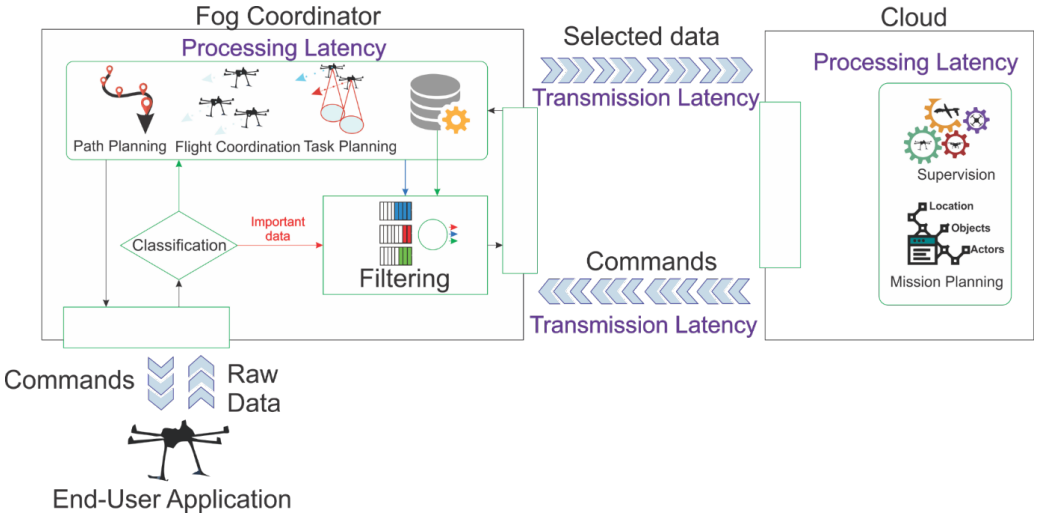
The services specified in the last topic can be deployed to the nodes using different methods ranging from virtualization of components (i.e. containing apps and libraries) to direct programming of the services in the host operating system. The selected system-on-a-chip (SoC) should be power efficient and support the fog services and the employed methods. Several components can be applied to this task and their common characteristics are presented in Table 10.

4.4 PROPOSED FRAMEWORK

Figure 20 details this approach. The idea is based on [104], [110] and [111]. Some of the data is sent to the supervision devices such as images and the geographical positions. However, all data is marked according to the type of message. The data coming from the different aircraft is received in the coordinator and classified based on their importance. Posteriorly, they are sent either to local processing or to filtering. If classified as important, the data goes directly to the filtering block to be clustered and sent directly to the GS. Otherwise, the data is transferred to local processing, which can process or store. This means that the coordinator has similar algorithms to the ones employed on the GS, i.e., the fog level can assist the UAVs during the execution of their tasks. Image and other data can also be stored at this level to be recovered by the GS in the future.

Figure 23 also shows the latencies considered in the model.The execution of the tasks available in the head coordinator along with the data classification and filtering will result in the processing latency at the fog level. The transmission latency is a function of the channel throughput and its characteristics. The processing latency in the cloud is a result of the services execution time provided by the cloud data centers.

Figure 23 – Fog-cloud computing network dataflow.



The comprehension of the dataflow can be improved by analyzing two SAR examples. The first one is about non-important data during the mission. In this example, the UAVs should regularly update their positions to enable continuously tracking by the GS. As this regularly updating can account for a large amount of data over time, the local processing can analyze if the position has changed enough to justify this action. Another example is the fog device taking actions locally and informing the GS in case that the aircraft is lost or did not update the position regularly. In SAR and Inspection missions, the recognition of a target is an important data that should be informed to the GS immediately.

4.5 FEASIBILITY MODELLING

4.5.1 SYSTEM MODEL

Few considerations are required to model the system. First, the fog device physical constraints limit the processing. Thus, the workload l_i is allocated by the processing capability of the fog v_i^{fog}. Other limitations are the fog-cloud communication stability and the average throughput required by the fog device that should be lower than network throughput.

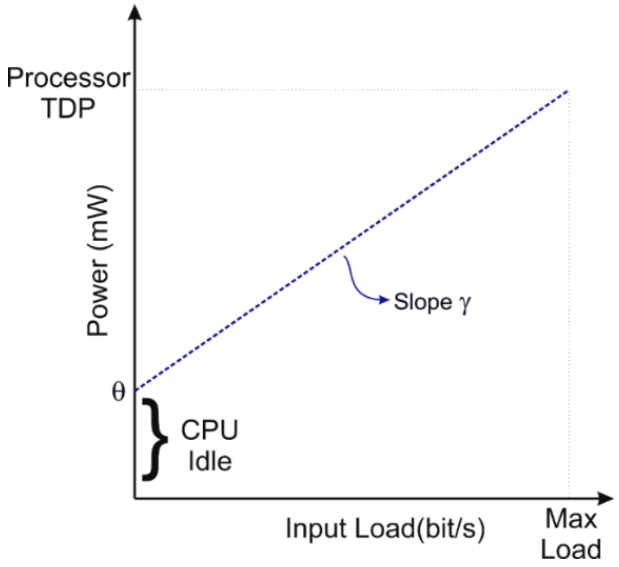
4.5.2 SYSTEM MODEL

Throughput: For a fog-cloud computing collaboration, the throughput B_i^{fog} can be modelled as a function of the average arrival rate δ_i^{fog}, the average packet size s_i^{fog} and the rate of packets rejected for fog processing $(1 - r_i)$. Note that the physical limitations of the system may affect the throughput of the communication, such as the available processing power and the end-device behavior.

$$B_i^{fog} = \delta_i^{fog} \cdot (1 - r_i) \cdot s_i^{fog} \tag{4.1}$$

Power consumption: Several factors may influence the power consumption in fog and cloud. These factors include the used algorithms and the environment temperature. However, the main factor is the computational load due to the information coming from the end-devices. For simplification, the proposed model considers the rate of data accepted for processing (i.e. Bits/second) and uses a standard value for the idle power consumption. This is shown in Figure 24.

Figure 24 – Power consumption at the fog device.



Then, the power consumption of a fog device i is modelled as a linear function of the data accepted for processing rate r_i in respect with the pre-determined processing efficiency $\gamma > 0$ and the idle power usage $\theta > 0$. The consumption boundary is mainly determined by the processor Thermal Design Power (TDP) or $E_i^{fog} \leqslant TDP$.

$$E_i^{fog} = \gamma \cdot \delta_i^{fog} \cdot s_i^{fog} \cdot (r_i) + \theta \tag{4.2}$$

Note that the power usage at the cloud is not considered. This is to ensure a power consumption optimization at the fog level, which will result in increasing flight time for the aircraft. This equation captures the amount of information selected for local processing. Since the most relevant information is desired to only be processed locally, thus, minimizing the energy consumption will penalize the local data processing to ensure this condition.

Latency computation of fog-cloud computing: The latency ω_i^{fog} in the fog device should be lower when few packets are transferred to the cloud and should grow exponentially if the number of packets exceeds the processing capability v_i^{fog}. The process should be stable if the average arrival rate δ_i^{fog} at the fog level is lower than its processing capability.

$$\omega_i^{fog} = 2^{\frac{(\delta_i^{fog} \cdot r_i)}{(v_i^{fog})}} \tag{4.3}$$

If the average is selected to contain only stable arrival rates ($\delta_i^{fog} < v_i^{fog}$), the function can be approximated as:

$$\omega_i^{fog} \approx \frac{(\delta_i^{fog} \cdot r_i)}{(v_i^{fog})} \tag{4.4}$$

The latency at the cloud ω_i^{cloud} is represented by the relationship between the data that is transmitted to it, the network throughput and the processing capability of the cloud data center. The amount of data sent to the cloud can be determined by the following function $s_i^{fog} \cdot \delta_i^{fog} \cdot (1 - r_i) = d_i^{cloud}$ and the amount of data transmitted between fog and cloud devices by $t_i^{(fog-cloud)}$. After being processed at the cloud, a fraction of the input data a_i^{cloud} is sent back to the fog. The perceived latency computed from cloud data is detailed in the below equation.

$$\omega_i^{cloud} \approx \frac{d_i^{cloud}}{2 \cdot t_i^{(fog-cloud)}} + \frac{d_i^{cloud} \cdot a_i^{cloud)}}{2 \cdot t_i^{(fog-cloud)}} + \frac{d_i^{cloud}}{2 \cdot v_i^{cloud}} \tag{4.5}$$

Where v_i^{cloud} represents the processing capability of the cloud. The total latency is given by the equation presented below, where ω^{fog} and ω^{cloud} denotes the respective latency in the fog and the cloud.

$$D_i^{AVG} = AVG(\omega^{fog}, \omega_i^{cloud}) \tag{4.6}$$

Two state parameters are defined to simplify the problem. The first one $x_1 = \delta_i^{fog} \cdot r_i$ represents the total quantity of packages in the fog device and the second parameter $x_2 = \delta_i^{fog} \cdot (1 - r_i)$ is the total of packages in the cloud. This problem can be re-written to minimize three objective functions:

$$min[B_i^{fog}, E_i^{fog}, D_i^{AVG}]^T = \begin{bmatrix} 0 & s_i^{fog} \\ \gamma & 0 \\ \frac{1}{2} \cdot v_i^{fog} & \frac{t_i^{(fog-cloud)} + v_i^{cloud}}{2 \cdot t_i^{(fog-cloud)} \cdot v_i^{cloud}} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \theta \\ 0 \end{bmatrix} \tag{4.7}$$

As can be seen, the model evaluates the tradeoff among bandwidth, energy in the fog node and the delay perceived by the UAV. This model should capture the variables behavior for a given set of parameters from the application when using the proposed architecture. Moreover, the goal of reducing latency forces an amount of data to be accepted for local processing in the fog computing node. As this variable grows, the energy constraints should keep this balanced within an acceptable boundary due to local processing capability and energy consumption.

4.5.2.1 MODIFICATION 1

The previous model represents one of the possibilities for analyzing the fog-cloud computing viability. However, other few considerations can be also relevant depending on the

hardware requirement. One important assumption is to consider the power consumption almost linear with respect to data processed in the fog device. However, this may not be true in every case. The power consumption required to transmit data to the cloud can also be a relevant part of the total power used in the head coordinator. Thus, a modification using an extra parameter "ρ"to represent the power required for data transmission is shown in:

$$E_i^{fog} = \gamma \cdot \delta_i^{fog} \cdot s_i^{fog} \cdot (r_i) + \rho \cdot \delta_i^{fog} \cdot s_i^{fog} \cdot (1 - r_i) + \theta \tag{4.8}$$

4.5.2.2 MODIFICATION 2

The deterministic comportment of the transmission latency is an important feature for analyzing the model behavior in the experiments. However, in some situations, it may be useful to analyze the latency behavior in a more realistic state. The modification of the model includes a random latency component $f(d_i^{cloud}, t_i^{(fog-cloud)})$ with a mean value equal to 1 and limited variance. The next equation presents the variation.

$$\omega_i^{cloud} \approx \frac{d_i^{cloud}}{2 \cdot t_i^{(fog-cloud)}} \cdot f d_i^{cloud}, t_i^{(fog-cloud)} + d_i^{cloud} \cdot \frac{a_i^{cloud}}{2 \cdot t_i^{(cloud-fog)}} + \frac{d_i^{cloud}}{2 \cdot v_i^{cloud}} \tag{4.9}$$

4.6 PARTIAL CONCLUSIONS

This chapter introduced a model to investigate fog-cloud computing cooperation to overcome the throughput and latency involving multiple aircraft in areas with low communication infrastructure. Besides, the analyzed model includes the power consumption limitations of UAVs. The developed fog-cloud scheme assigns a UAV as head coordinator in a fog computing level to process and to control the communications between the nearest nodes and the cloud. The UAV coordinator manages the computational offloading between cloud and fog. Besides, this coordinator enables the continuity of the autonomous operation even when it is not connected to the ground station. This process results in a lower latency when dealing with the real-time data provided by the aircraft.

This proposed model differs from previous works by capturing performance behavior not totally explored in UAV scenarios including the limited amount of energy, processing capability restrictions at the fog computing and limited throughput among fog and cloud computing. Those characteristics may also be found in other applications (e.g., IoT) which makes the model even more interesting.

5 ENVIRONMENT AND EXPERIMENTAL RESULTS

This section describes a few experiments as a representative example of ARCog. The output environments are based on three kinds of missions: (1) search and rescue; (2) inspection; (3) and surveillance. The results aim at demonstrating the main components of the ARCog once they are essential to a fully automated operation of a UAV. The main objectives of the missions can be summarized as follows:

- *Testing the capability of self-localization while performing the mapping of the target position in a pre-determined areaa.*
- *Identifying targets and informing the human operator.*
- *Determining the final point of the mission when a pre-determined goal is found.*
- *Following the security requirements imposed on the system during the tasks execution and re-planning the mission when is necessary;*

5.1 MAIN ALGORITHMS

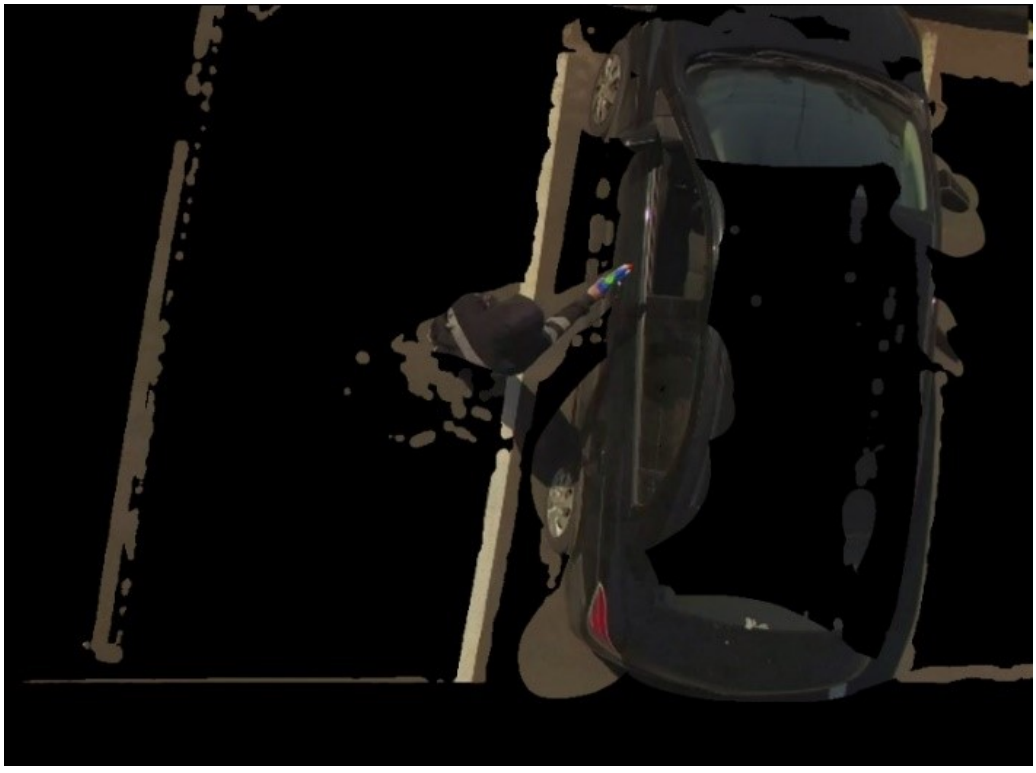
5.1.1 PRE-PROCESSING STAGE

The background removal is a necessary task to the pre-processing stage. Figure 25 has four screens showing the processing stages. The first screen (i.e., console image) shows the percentage of pixels that were removed from the original image. Screen 2 presents the output data. As can be seen, only the moving parts of the image are recognized. The original image is on screen 3. The green dots are the automatic selection of pixels necessary to perform the Lucas-Kanade method. These points are chosen using OpenCV function *goodFeaturesToTrack* that selects the high contour, i.e., corners in the image. Screen 4 shows the image transformation through the homography approach based on the movement of the selected pixels between two images.

Figure 25 – Aerial background removal test.

Figure 26 presents another example of an output image with a mask generated by the background removal algorithm. The removal percentage varies drastically according to the number of moving objects in each image. In this image, the algorithm was able to surpass 60% of the original image.

Figure 26 – Example of the mask applied by the background removal algorithm. The scene corresponds to a man holding a gun while approaching a car.

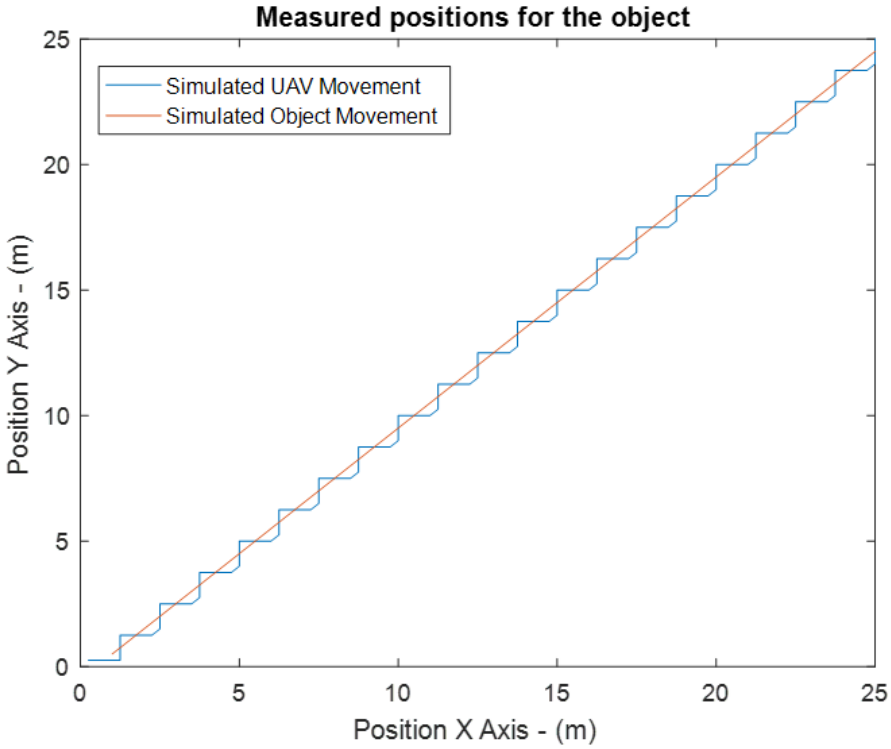


5.1.2 UAV AND TARGET TRAJECTORY PREDICTION

Some simulations and experiments were performed to validate the system operation. The covered scenarios were chosen to comply the following conditions: (1) UAV in a static position with object moving; (2) Object in a static position with the UAV moving; (3) Both object and UAV moving (also including occlusion).

Figure 27 presents the positions simulated for the UAV and the target object. Note that both of them are moving. As can be seen, the UAV performs a zigzag movement while the object is moving in a straight line. This pattern was chosen because the combined movement between the parts results in a change of only a few meters in each axis (i.e., approximately 1.25 meters), right in the range of the added signal error, i.e., the amplitude difference between red and blue signals have the same mean values that the signal error added in the last step. As a result, the perceived object position can be easily mistaken with the error. This scenario represents the worst case for the filter.

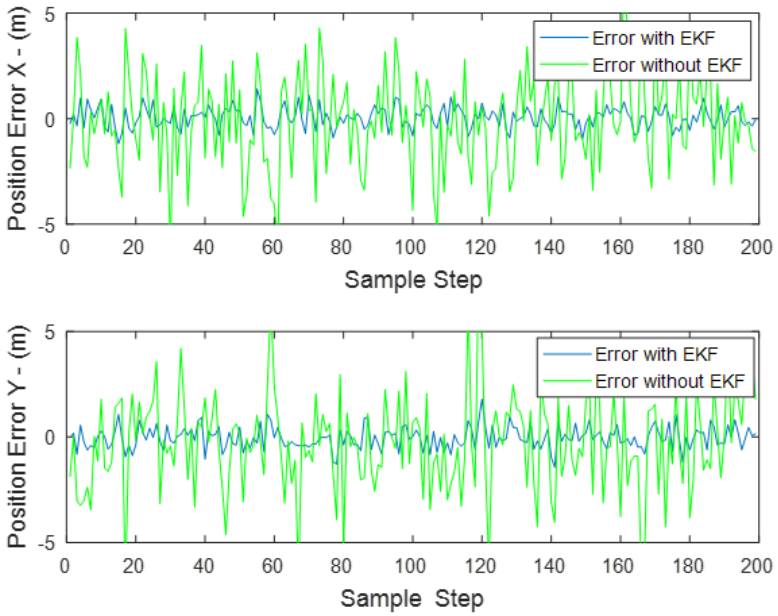
Figure 27 – Simulated positions for UAV and object.



This presented pattern is used to generate the data applied to the algorithm. The positions are derived for creating accelerations (i.e., linear and angular). They were feed in the algorithm as an incremental GPS position. Before the test, all data was corrupted by 10% of Gaussian white noise. This was chosen by the authors as a way to test the CBR under errors condition. Note that Figure 28 presents a comparison between the positions errors from the real object that is given by the difference between the pattern information and the corrupted information. Note that the error using EKF is lower than the other about 30% of mean error and with standard

deviation improvement of 23.2%. The second and third scenarios had smaller gains than the first one. This is possibly related to the amount of noise, which can be explained by the fact that the UAV or the object is stationary. Thus, the errors of the positions are close to zero, i,e, the error information was not added to the aircraft or the object. The improvement stayed in the range of 23% mean error with standard deviation improvement of 45%.

Figure 28 – Position errors compared to the pattern information.



For the experimental tests, the UAV movement was moving in a predefined trajectory at a height of approximately 10 meters. The tracked object was kept in a fixed position on the ground. The exact UAV pose in the air is not known (i.e., ground truth is unknown) and this not allows a proper calculation of its movement error. However, as the object position is in a fixed place on the ground, the position perceived by the UAV would not change if the system error was null. Figure 29 shows the experiment schematic. Note that error can be exactly calculated at each moment through direct measurement of the filter output of the object position variation.

Figure 29 – Schematic of the EKF testing.

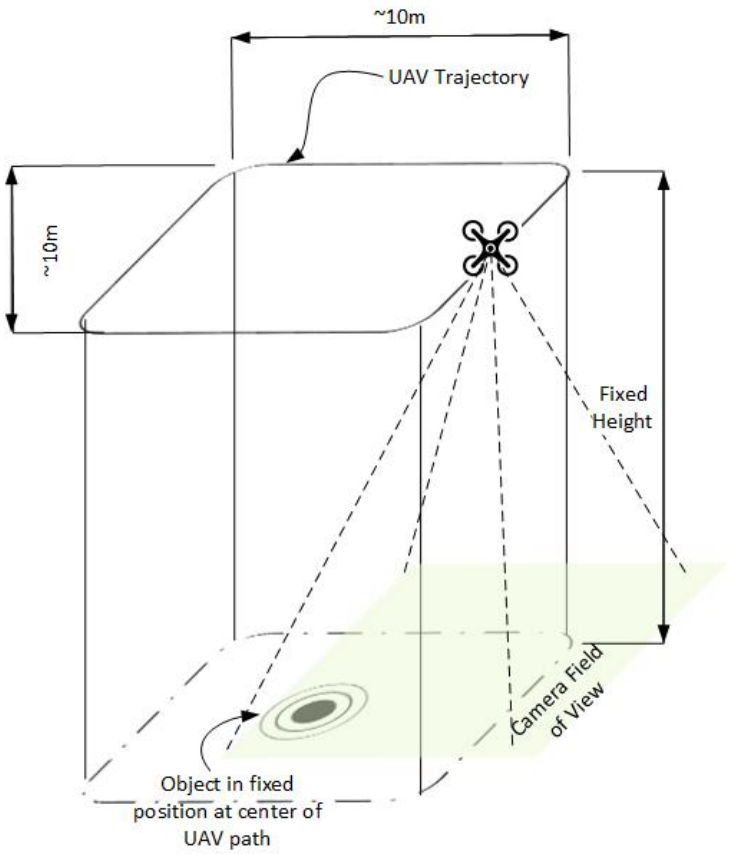
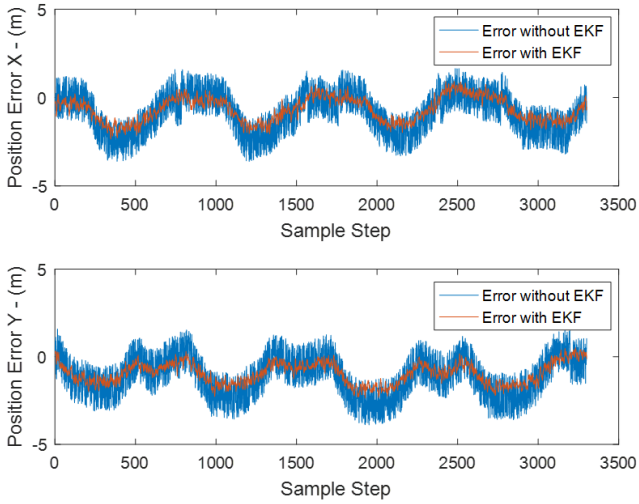


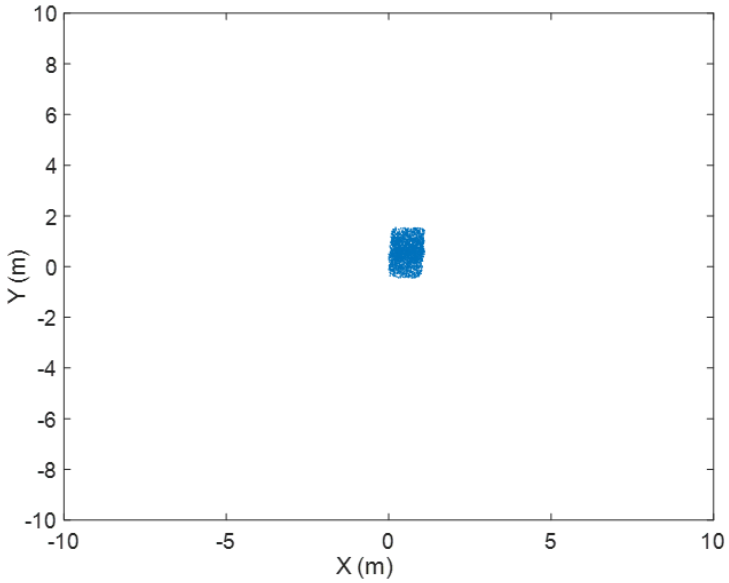
Figure 30 shows the error curve in each axis for the real-world experiment of the object location estimation with and without the Kalman filter. There is an improvement of 21% in the mean error and 30% in the standard deviation. The error in this situation can be directly linked to GPS as the main factor. Most of the time, the error stayed below one meter in each axis, representing a good error value concerning the GPS capability.

Figure 30 – Object position errors compared to the real location



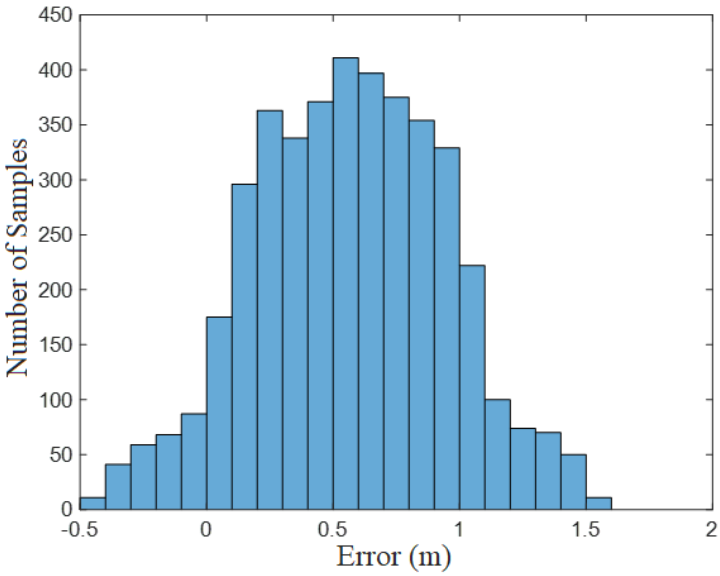
The filter capacity was evaluated to predict a given object position that is in a fixed point while the UAV is moving within a predefined trajectory to test the last part of the Kalman filter. When the object is fixed on a determined position, any prediction performed by the UAV represents directly the calculation error, which makes easier the perception of the position prediction error. In the experiment, the UAV moved within a square of approximately 10 meters with the object always kept in the camera field of view (FOV). Figure 31 shows the calculated object position in each measurement with a sampling rate of approximately 10 frames/second during the image capture.

Figure 31 – Calculated object position in each measurement of the UAV.



The mean error obtained was 0.56 meters with a standard deviation of 0.4. Figure 32 presents the object position error histogram. This error has a more centralized distribution of around 0.5 meters. This result is possibly due to the combination between the algorithm position error that calculates the UAV pose with the calculation errors of the object position.

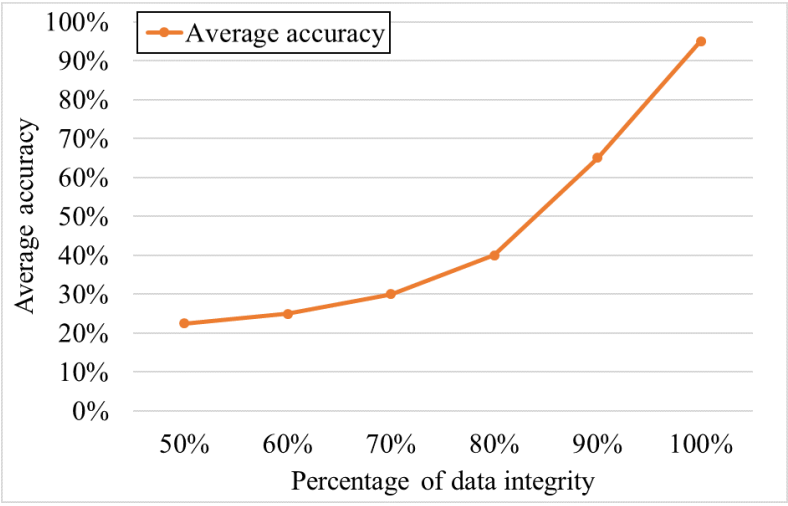
Figure 32 – Histogram of the object position error.



5.1.3 CBR EVALUATION

For this test, ten cases were collected. They were edited by randomly removing some fields from the input cases to enable the simulation of the partial data coming from OpenCV. Besides, they were rated by the percentage of information (i.e., original instance parameters) presented from the original case, which is the input case. Figure 33 presents the accuracy output of the system concerning the percentage of data integrity. Note that the accuracy decays as more incomplete the cases information becomes.

Figure 33 – CBR accuracy.

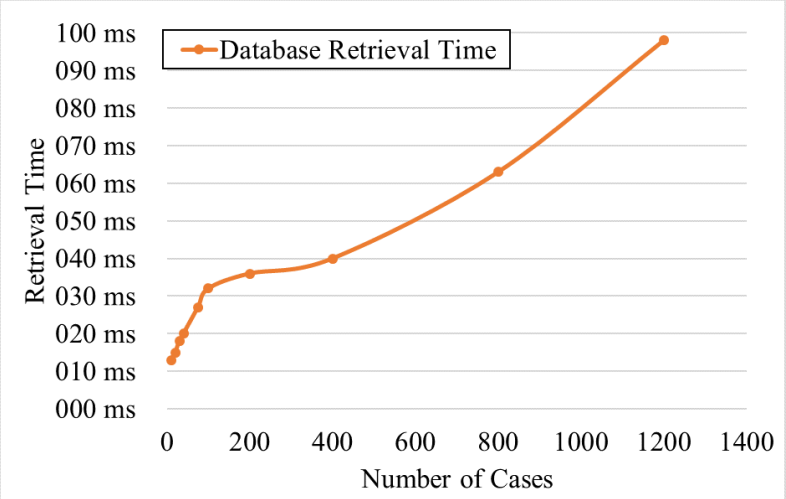


The case retrieval time was calculated for a set of cases containing different inputs in the database aiming at the evaluation of the CBR process scalability concerning the possibility of case growth during the operation of the system. The result of the retrieval time calculation in 5.1 is the sum of the following components: (a) search time in the database (T_S) that is proportional

to the number of cases; (b) the similarity calculation time (T_C), which is proportional to the number of fields analyzed in the case; (c) the time related to query formation (T_Q); (d) the time to prepare the display for the user (T_O). This experiment tested 10 queries to verify system performance. Note that the worst case was selected (i.e., a case with high retrieval time). Figure 34 shows the results of the retrieval time. They were measured using the equation presented in 5.1. The test was performed by searching the worst case against a database with size varying between 10 and 1200 cases. Note that the overall tendency of the response time can be approximated for a 1st-degree polynomial, which indicates the scalability of the algorithm. Note that the response time of 100 milliseconds is lower than other kinds of delays, e.g., the network delay, even with a massive amount of cases (i.e., 1200).

$$T_F = \max_{i=1:10} \sum (T_S + T_C + T_Q + T_0) \tag{5.1}$$

Figure 34 – CBR retrieval time.



5.2 SURVEILLANCE MISSION

Figure 35 shows the UAV used in the experiments of surveillance and SAR missions. The architecture was tested using an onboard computer equipped with an Intel atom z8785G with 4 cores and 2 GB of RAM. This equipment was chosen due to its low power consumption, and fanless operation capability. The centralized computer is the i7 7700hq with 8GB of RAM. The UAV has a Logitech full HD camera, a Pixhawk 2.4.8 32-bit flight controller with an ArduPilot firmware and NEO-8M GPS. Table 12 presents the collected scenes for the surveillance mission.

The prediction results of the classes Car, Person and Gun are evaluated using a confusion matrix to test the ability of the recognition algorithm. The confusion matrix is considered as an important estimation technique to evaluate the prediction results coming from remote sensors [112]. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The basic terminology is:

Figure 35 – UAV used in the surveillance and SAR missions.

Table 11 – Collected scenes for the surveillance mission.

Scenes	Car	People	Objects	Event
1,2,3	Absent	Present	Gun	Theft
4,5	Present	Absent	Gun	Theft
6,7	Absent	Present	Spray	Graffiti
8,9	Present	Absent	Absent	Irregular Parking
10,11	Present	Present	Absent	People/Car Interaction
12,13	Absent	Present	Absent	People Interaction

- **Condition Positive (P):** The number of real positive cases in the data;
- **Condition Negative (N):** The number of real negative cases in the data;
- **True Positive (TP):** Equivalent with hit;
- **True Negative (TN):** Equivalent with correct rejection;
- **False Positive (FP):** Equivalent with false alarm;
- **false negative (FN):** Equivalent with miss;

Table 12 shows the evaluation parameters of the confusion matrix. Table 13 shows the results of the classes "Car", "People"and "Gun". Table 14 shows the parameters obtained through the use of the confusion matrices, such as the accuracy of the detection and the recognition error distribution in the chosen classes. Besides, it also considers likelihood ratios LR+ and LR-. Note that the recognition sensitivity of the class "Cars"is greater than the other classes (i.e., "People"and "Gun) due to the size of the car concerning the resolution of the image results in a greater amount of sharp details.

Our results can be compared to other related works. A simplest approach is presented in [113], where the algorithm only recognizes the presence or the absence of humans in a scene.

Table 12 – Confusion Matrix Parameters.

TPR	$\frac{\Sigma TruePositive}{\Sigma ConditionPositive}$	$\frac{TP}{TP+FN}$
FNR	$\frac{\Sigma TrueNegative}{\Sigma ConditionNegative}$	$\frac{TN}{TN+FP}$
PPV	$\frac{\Sigma TruePositive}{\Sigma PredictedconditionPositive}$	$\frac{TP}{TP+FP}$
NPV	$\frac{\Sigma TrueNegative}{\Sigma PredictedConditionNegative}$	$\frac{TN}{TN+FN}$
FNR	$\frac{\Sigma FalseNegative}{\Sigma ConditionPositive}$	FP+TN=1-TPR
FPR	$\frac{\Sigma FalsePositive}{\Sigma ConditionNegative}$	FP+TN=1-TNR
FDR	$\frac{\Sigma FalsePositive}{\Sigma PredictedConditionPositive}$	FP+TP=1-PPV
FOR	$\frac{\Sigma FalseNegative}{\Sigma PredictedConditionNegative}$	FN+TN=1-NP
ACC	$\frac{\Sigma TruePositive+\Sigma TrueNegative}{\Sigma TotalPopulation}$	$\frac{TP+TN}{TP+TN+FP+FN}$
Prevalence	$\frac{\Sigma ConditionPositive}{\Sigma TotalPopulation}$	$\frac{TP+TN}{TP+TN+FP+FN}$

Table 13 – Class car (Instances: 216) / Class People. (Instances: 247). / Class Gun. (Instances: 270)

Class Car		Positive	Negative
Predicted Class	Positive	100	5
	Negative	0	111

Class People		Positive	Negative
Predicted Class	Positive	67	2
	Negative	34	144

Class Gun		Positive	Negative
Predicted Class	Positive	44	13
	Negative	46	167

Table 14 – Results of the confusion matrix parameters.

Parameters	**Class: Car**	**Class: People**	**Class: Gun**
ACC	0.98	0.85	0.78
FDR	0.05	0.03	0.23
NPV	1.00	0.81	0.78
DOR	0.00	0.19	0.22
LR+	23	48	6.8
LR-	0	0.3	0.6
TPR	1	0.66	0.48
FNR	0	0.33	0.51
FPR	0.04	0.01	0.07
TNR	0.96	0.99	0.93

The classification results are comparable with the ones showed in this research, and its better accuracy is the tradeoff with the simplistic implementation that is not capable of recognizing more objects or actions. The results presented in [114] have little more classes to be separated. The tracking activity is applied directly using BOW, which makes the architecture more rigid when adding new activities.

The proposed system showed improvement in perception accuracy. To evaluate the whole system operation, the complete algorithm was applied to the data collected in Table 16. For each video, a segment comprised of 180 frames was selected. The high-level decision maker analyzed this segment against the expected result considering two classes: when the event is present (true positive) and when the event is not presented (true negative). Table 15 shows the CBR decision accuracy. It is possible to note that the high-level results detection are aligned with the low-level ones.

Table 15 – Event's detected for each occurrence.

Class	**Decision Accuracy**
Predicted Theft	
True Positive	70 %
True Negative	95 %
Predicted Vandalism	
True Positive	65 %
True Negative	93 %
Predicted Driving Violation	
True Positive	92 %
True Negative	95 %

5.3 SEARCH AND RESCUE MISSION

For the SAR mission, the ARCog was tested using the same UAV presented in the surveillance mission. The communication among the UAVs and the GS was made over Wi-Fi to simplify the implementation and development. However, this technology can be replaced by other methods to meet the requirements of specific applications.

As the aircraft localization relies on GPS information, the object localization does not require high accuracy. Thus, even a few meters of error allow rescue teams to localize the objects. The environment understanding is supported by visual perception through the measurements of the image recognition process. A full HD camera Logitech c920 is applied to help this application. In case that the system is not capable of processing high resolutions images, the downscaling proceeding will improve sharpness and contrast.

A few tests were conducted to evaluate the HMI. The proposed scenarios were designed to assess the system capacity for responding in different conditions. The experiments were

Table 16 – Scenes for the SAR mission.

Scenes	Car	People	Expected Action
1	Present	Present	Inform
2,3	Present	Absent	Register
4,5	Medium	Present	Inform
6	Absent	Absent	None
7	Absent	Present	Land

performed with the UAV flying in a predefined trajectory at a height of approximately 6-10 meters. Figure 36 illustrates the mission execution and path followed by the aircraft during the execution of the mission.

Figure 36 – Mission execution and trajectory followed by the aerial robotic system in the entire search and rescue mission.

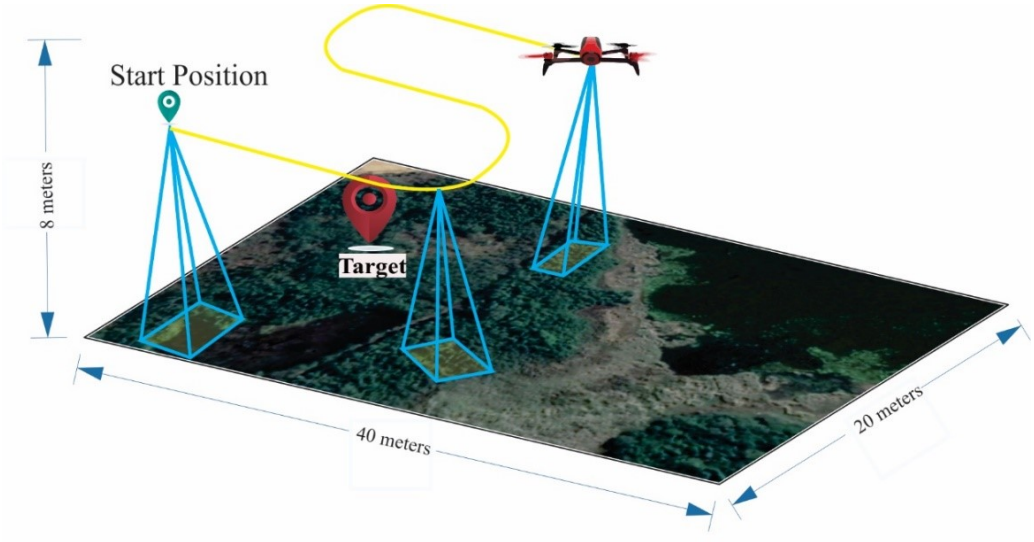


Table 16 presents the expected system actions in 7 scenarios. The data consists of the video recorded by the UAV along with the sensors' measurements. The HMI responses in these determined scenarios are available on https://github.com/ARCog/ARCog/wiki. All codes and interfaces were developed in a Linux compatible environment aiming at an easing integration with ROS. The rosbag containing the data from the UAV used in the video is available on the previous link.

Figure 37 details the HMI screen during scene 4 with the following presented information: (1) CBR analysis; (2) UAV flight details; (3) recorded videos, and images. In this figure, the HMI displays an image of scene 4 where the target is recognized. Besides, the decision making output is informing the GS of the identified object. The interface also assists the operator in visualizing the necessary data from the UAV and in receiving feedback about the system detection and new cases.

Figure 38 highlights the inter-block communication in the experiment among some of the associated ARCog blocks to exemplify architecture operation. This figure does not contain every

Figure 37 – HMI interface.

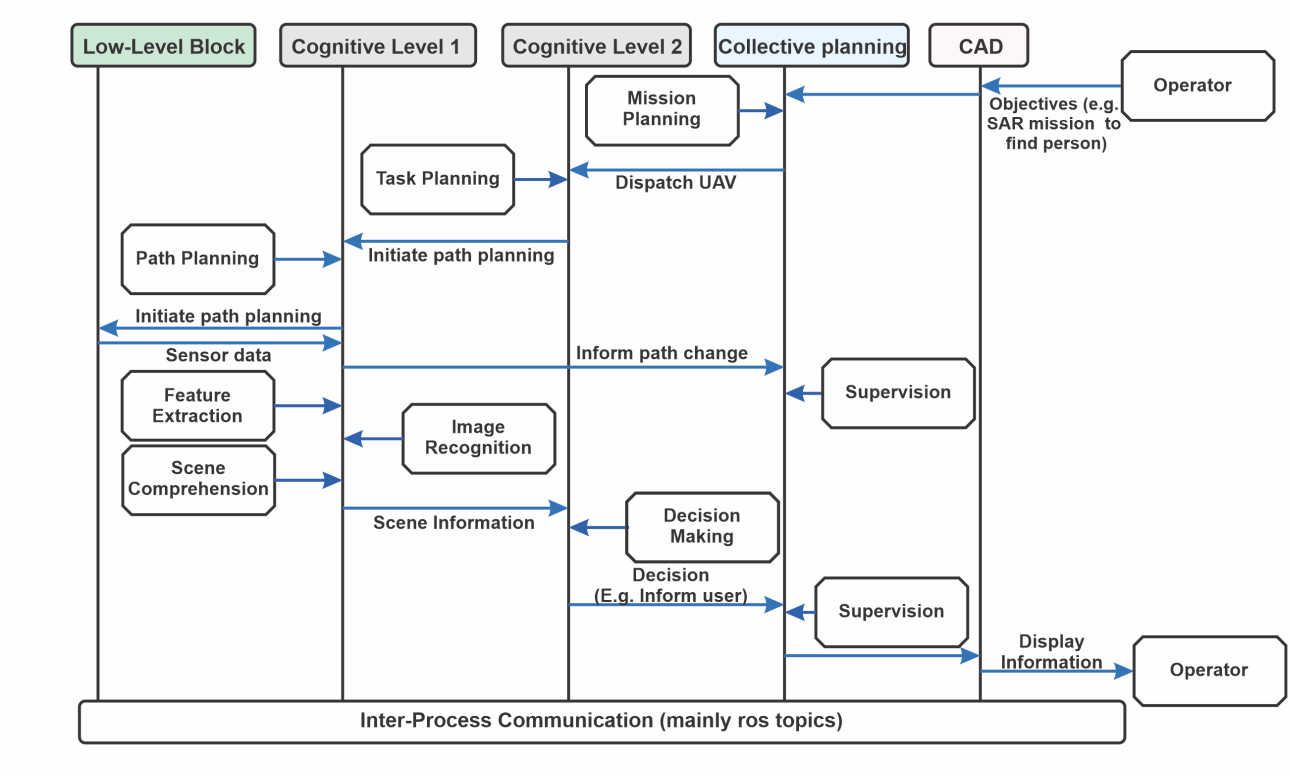
interaction due to the complex nature of the data flow. Instead, it only presents key interactions for illustrative purposes. This communication segment shows the mission start from HMI and the corresponding events subjected to the detection triggers in the decision-making process. It is possible to observe that during mission execution, several blocks are in parallel communication, such as the supervision and sensor data processing.

In this communication flow, object recognition has a vital role in the mission execution once it is responsible for determining the achievement of the mission goal. Thus, this algorithm performance has a significant influence on the SAR operation success. However, despite of its importance, this work does not propose improvements to it. The results related to the recognition algorithm should be seen as guidance for the system performance. For each video, a segment comprised of a few seconds was selected. In this data segment, high-level decision making was analyzed against the expected results. A positive value (true positive) in the decision is when the target is present and detected. However, when the target is not present, and its presence is not recognized, the event is negative (true negative). Table 17 shows this result.

Table 17 – Scenes for the SAR mission experiment.

Scenes	Target Detected	Decision Accuracy
Scene 1	Positive	90%
	Negative	95%
Scene 2	Positive	85%
	Negative	95%
Scene 3	Positive	86%
	Negative	93%

Figure 38 – Inter-block communication among the proposed architecture.



Another strong aspect of this kind of mission is the target location. Therefore, the UAV should record the positions where the target is recognized to ensure adequate planning to the ground rescue team when reaching the target. These next results were designed to show this behavior. Figure 39 exhibits the mission execution representation, and the trajectory followed by the aerial vehicle. Figure 40 presents the real data from scene 5, where the blue line is the UAV odometry during the flight, and the red one represents the part of the path that the target was recognized.

In some missions, the UAV could be in charge of delivering equipment to the target, such as the first aid kit. To show this action, a sequence of cases with the expected behavior were inserted in the decision making. Firstly, if the target was recognized with low certainty, the UAV had to change path planning and get closer to the goal to increase it. If the target was still identified and had higher recognition confidence, the UAV had to change path planning and land somewhere close to the target. Figure 41 shows this behavior and the key points where the decisions were taken. After landing, the UAV has to take decisions to take off and resume the mission.

To allow better understanding of the behavior described in Figure 41, a sequence of events was compiled to show the key points. Figure 42 illustrates these events. The recognition certainty, when the target was first recognized, was lower than 75%, then the UAV reduced its flight height and finally landed close to the target.

Figure 39 – Mission execution representation and trajectory followed by the aerial vehicle in a SAR mission.

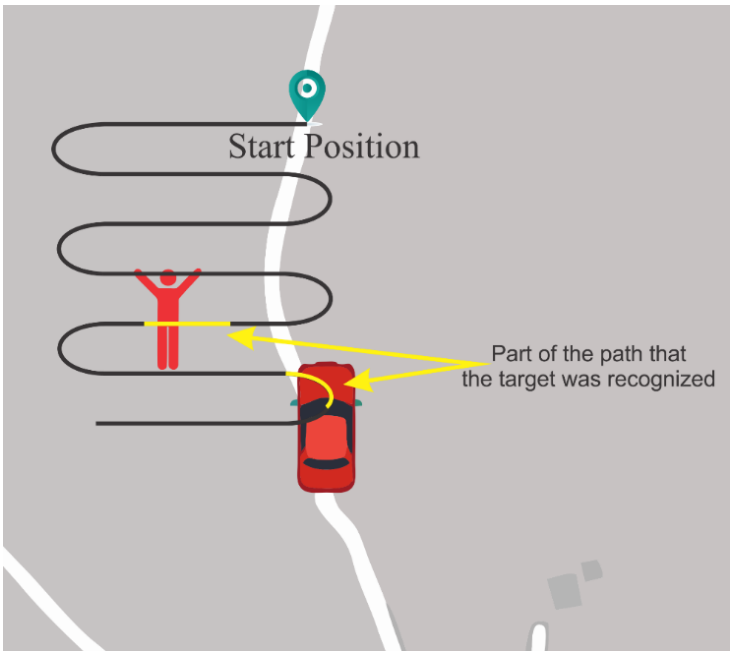
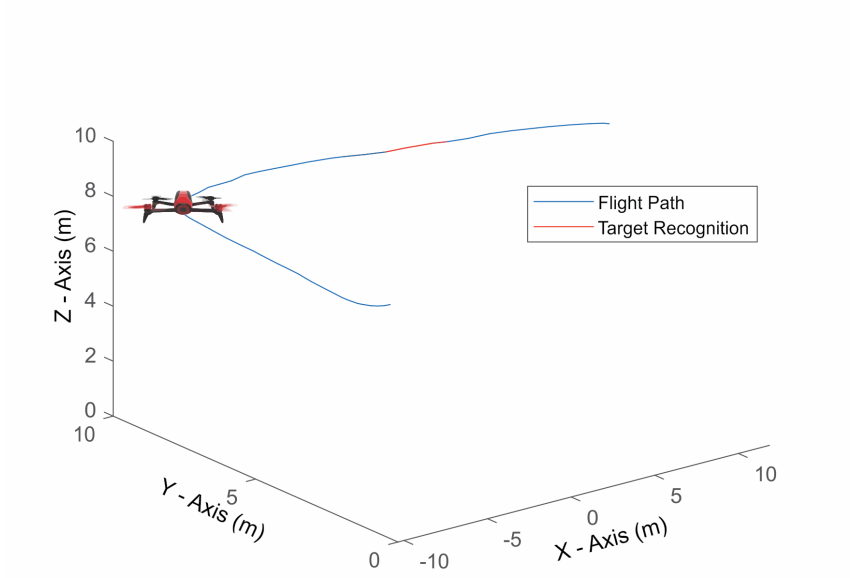


Figure 40 – Part of the real trajectory executed by the aerial vehicle in a SAR mission.



A few parameters of the software performance can be extracted during the mission, such as the ones in Table 18. They allow the understanding of the minimal processing power required by the onboard part of the ARCog. These values show that hardware improvements could benefit the system.

Figure 41 – ARCog decisions in a possible case of search and rescue mission.

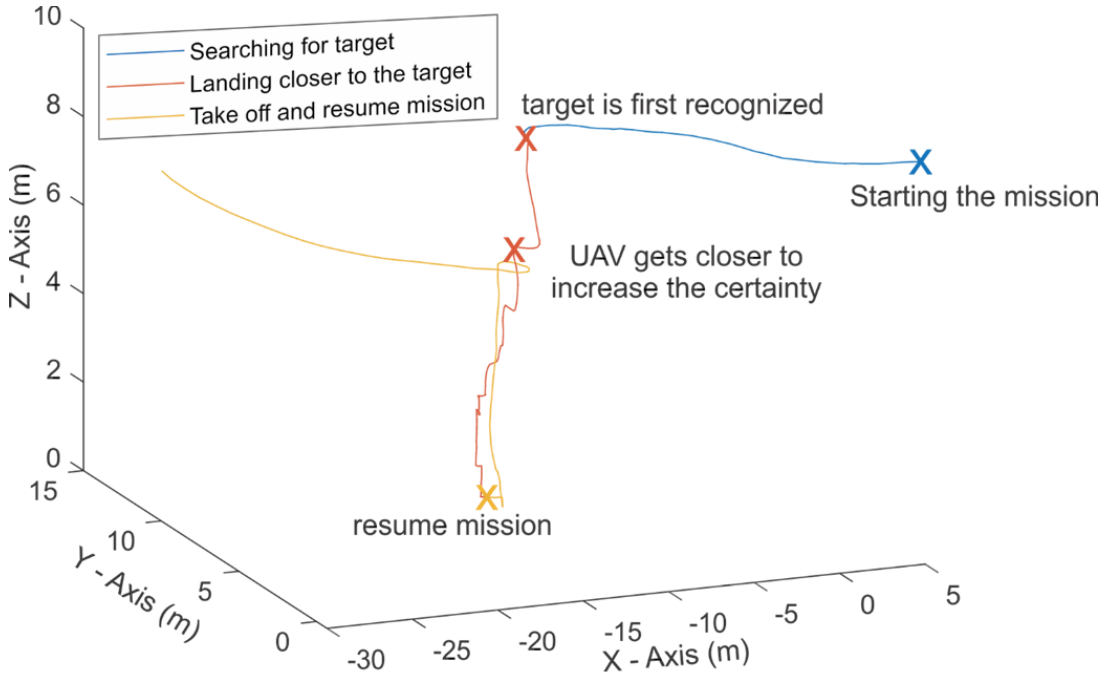


Figure 42 – Sequence of events in a possible case of search and rescue mission. (a) The target is recognize. (b) UAV approximates to the target aiming at increasing its recognition certainty. (c) UAV is preparing to land. (d) UAV landing.

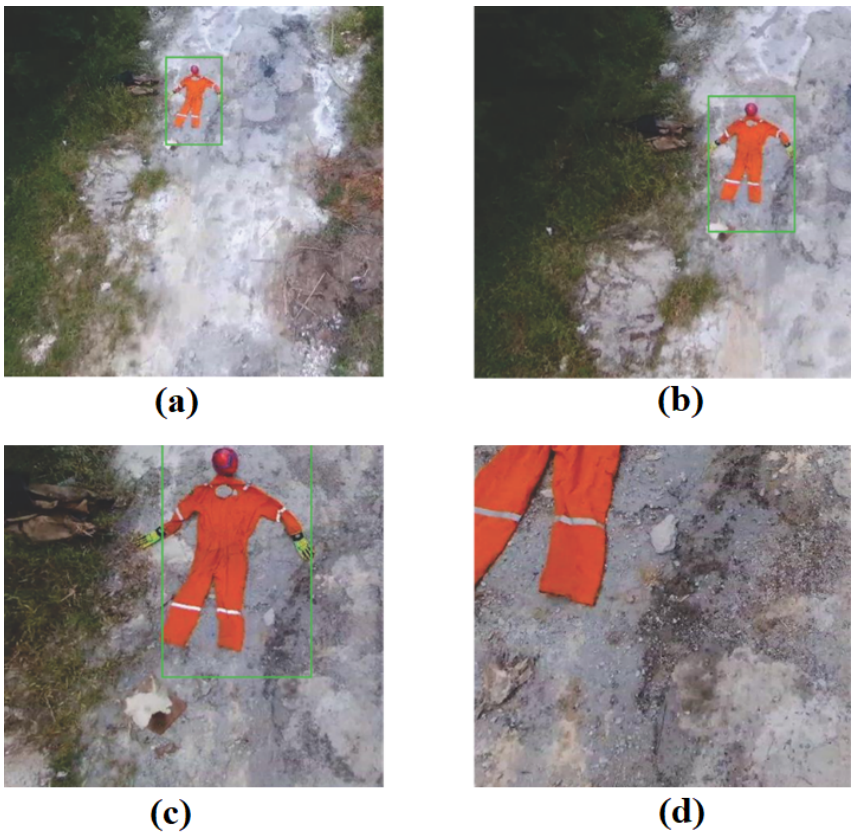


Table 18 – Software performance.

Parameter	Value
CPU Usage (%)	95%
RAM Usage (%)	85%
Mean number of ROS topics	31
Mean number of messages per second	136
Number of frames per second	2

5.4 INSPECTION MISSION

The ARCog in this mission was implemented using four different processing units. The first one is an A3 FCU board that runs the basic low-level control. The second is a companion board computer equipped with a quad-core, Nvidia ARM CPU and 4 GB of ram. This equipment was chosen due to its low power consumption and fanless operation capability, and it is used to run the low-level tactical activities. The first two boards are embedded on the Matrice UAV as shown in Figure 43. The third is a centralized laptop computer is the i7 7700hq with 8GB of RAM also used as a HMI. The final processing unit is also a laptop computer with an 8GB NVIDIA®GeForce®GRX1080 GPU. The architecture was built over the Ubuntu platform using ROS Kinect as a base for the development. The communication between the UAV and the ground station was built over Wi-Fi. The blue arrow indicates onboard communication, the dashed gray illustrates ROS topic-based message passing.

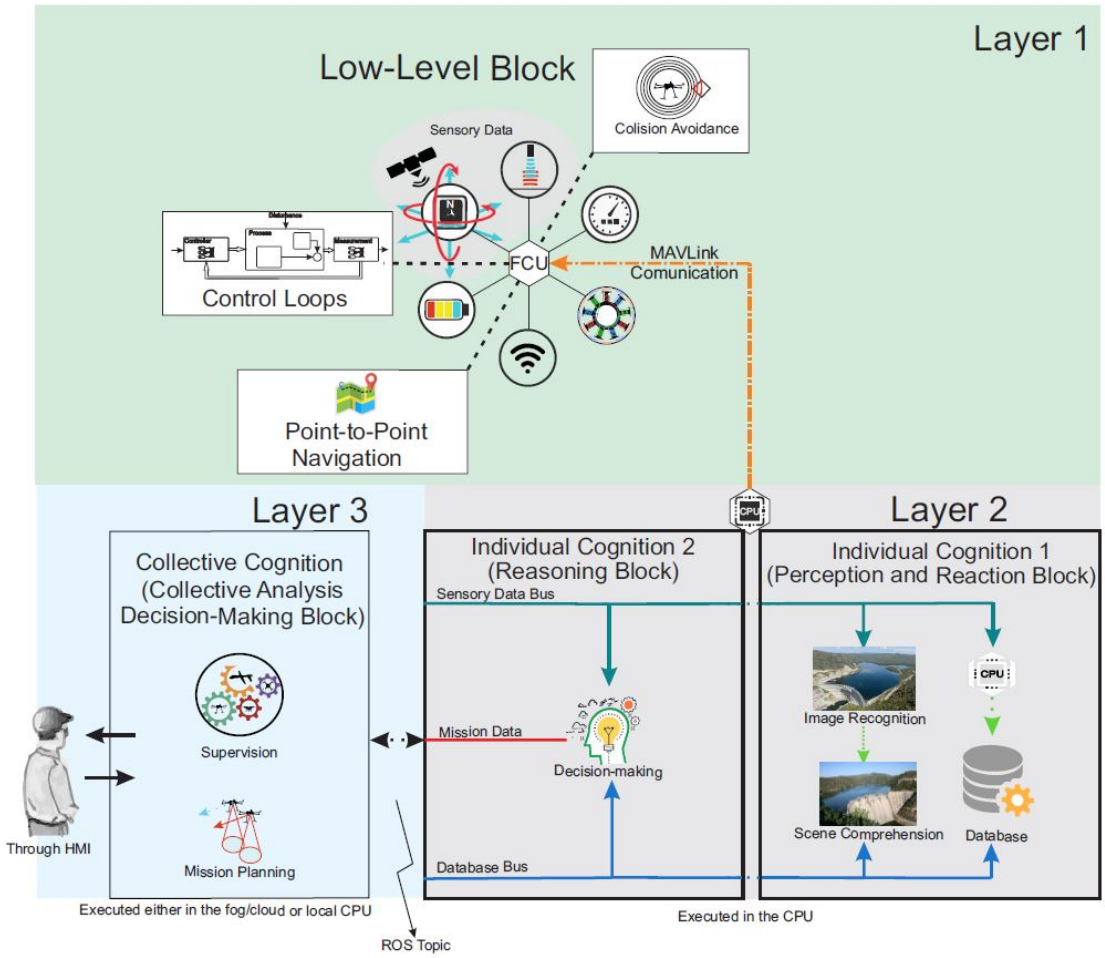
Figure 43 – UAV Matrice used as part of the ARCog test in the inspection mission.

The objective of the ARCog in this mission is to perform an inspection and to generate a detailed reconstruction of the area. The image processing is performed by the ground station that uses the photogrammetry technique to generate high precision spatial data from the acquired images. The visual perception supports the environment understanding through the measurements of the image recognition process. A full HD camera Zenmuse Z3 is applied to this application. The architecture evaluation was conducted to demonstrate the UAV autonomous operation to inspect for 3D reconstruction. The objective is to perform the inspection of interesting points and to structure these points for 3D reconstruction.

Due to the architecture flexibility of organizing, the ARCog was simplified for the inspection mission. Figure 44 exhibits a overview of the architecture layout composed of 5 mains blocks (including an HMI with a human-in-the-loop). Layer 3 contains the human-in-the-

loop unit once the inspection activity is simple. Besides, this layer will be responsible for the supervision and mission planning tasks.

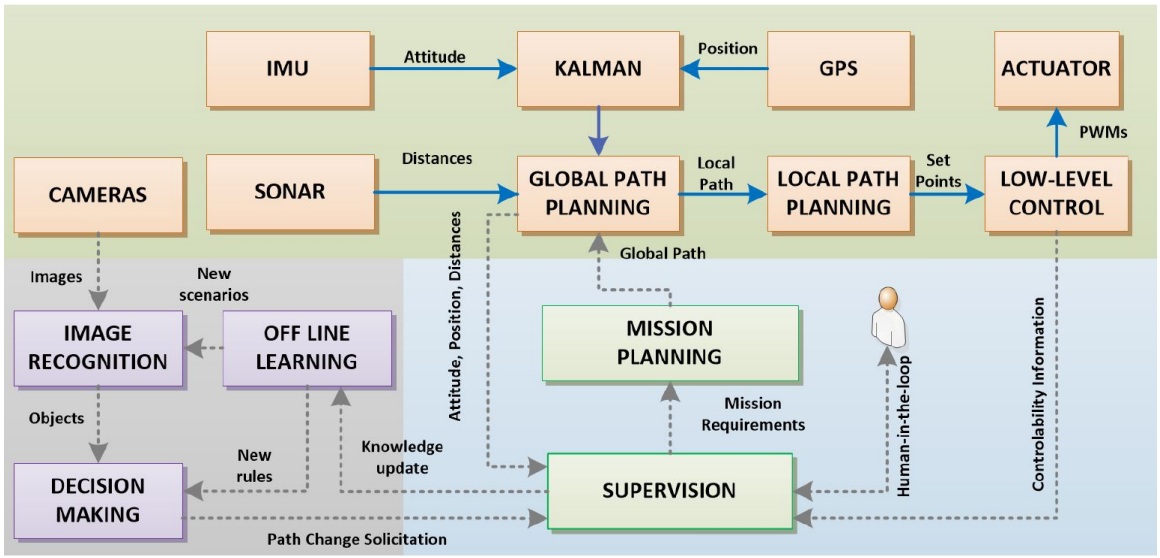
Figure 44 – Main components of the ARCog for the inspection mission.



The three first blocks (i.e., Low-level Block and Cognitive Levels 1 and 2) are the same that the expanded version of ARCog. The fourth block in layer 3 is responsible for the collective analysis of the multiple tasks inside the aircraft. This supervision block is usually encoded in a fog computing architecture when faster and more robust processing is required. For example, the UAV can send the acquired images to an external system for three-dimensional reconstruction. The main responsibilities are the generation of tasks and the supervision of the aircraft operation. Figure 45 describes the information flow inside all described layers, where the blue arrow indicates onboard communication; the dashed gray illustrate ROS topic-based message passing. The background color means where each block runs; green is the Low-Level Block and is processed by the aircraft's onboard FCU and companion board, blue is in the HMI laptop responsible for the cognitive decisions and the gray in the GPU laptop and controls the collective analysis. Therefore each level runs in a different unit.

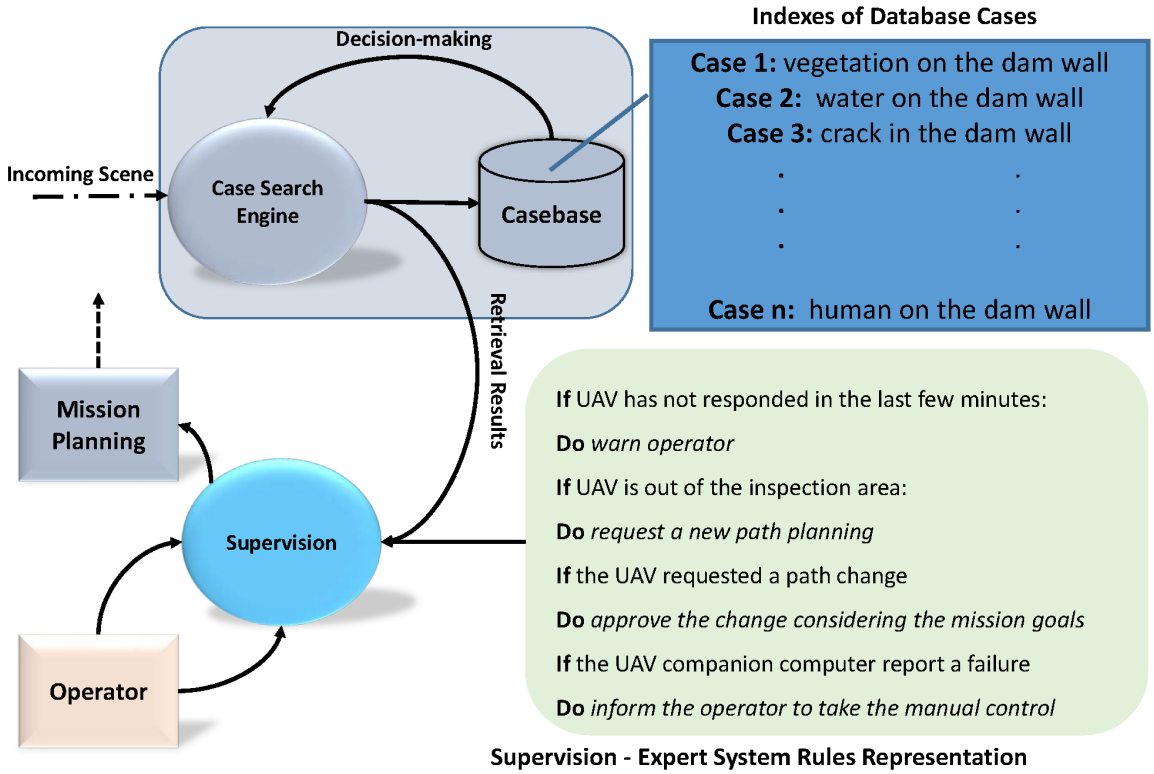
Figure 46 illustrates an example of the decision-making process and supervision. Note a few examples of cases that would be in the architecture decision. The CBR is responsible for the

Figure 45 – Communication Diagram showing the main components.



UAV related decisions such as the environmental stimulus reaction. The supervision holds an expert system in charge of assuring a properly UAV operation and that the onboard decisions are not compromising the mission or aircraft safety.

Figure 46 – Decision-making process and supervision.



In this research, the inspections are in two main areas. The first inspected area is an abandoned quarry. The aircraft has to perform a specific path over the vertical part of the area to inspect for loose parts. After, a horizontal path is required to obtain the area under the elevation. In this process, the system gathers enough information to allow a detailed reconstruction. Figure

47 shows the positions of the camera during the mission and the trajectory followed by the aerial robot during its execution. This pattern is usually applied to perform inspection and produces high coverage of the full area that will be analyzed. Figure 48 presents the reconstructed area as a final result. Note that the rocks positions can be accurately observed. This work uses a real-time 3D monocular reconstruction library named COLMAP to generate the dense point cloud [115] and analyze the pixel density and shapes. An initial analyzed and verified 3D reconstruction is used as ground-truth to find a 3D deformity. Then, the matching of common features between the current reconstruction and the ground-true is performed.

Figure 47 – Positions of the cameras during the inspection and the UAV trajectory.

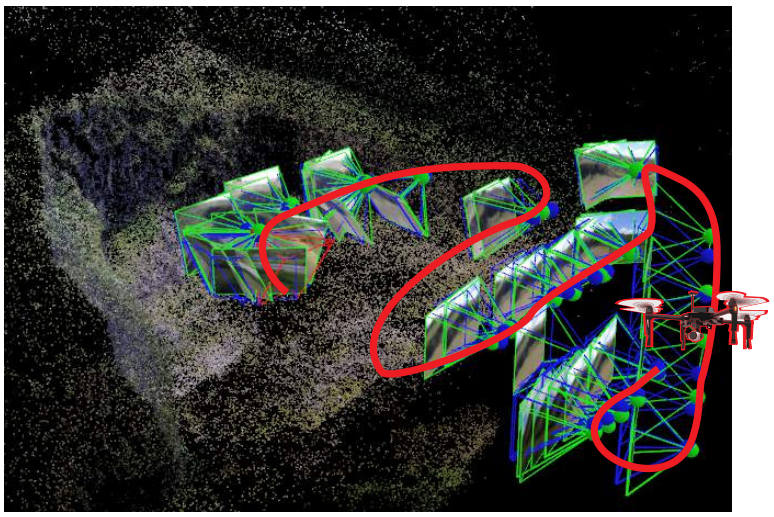
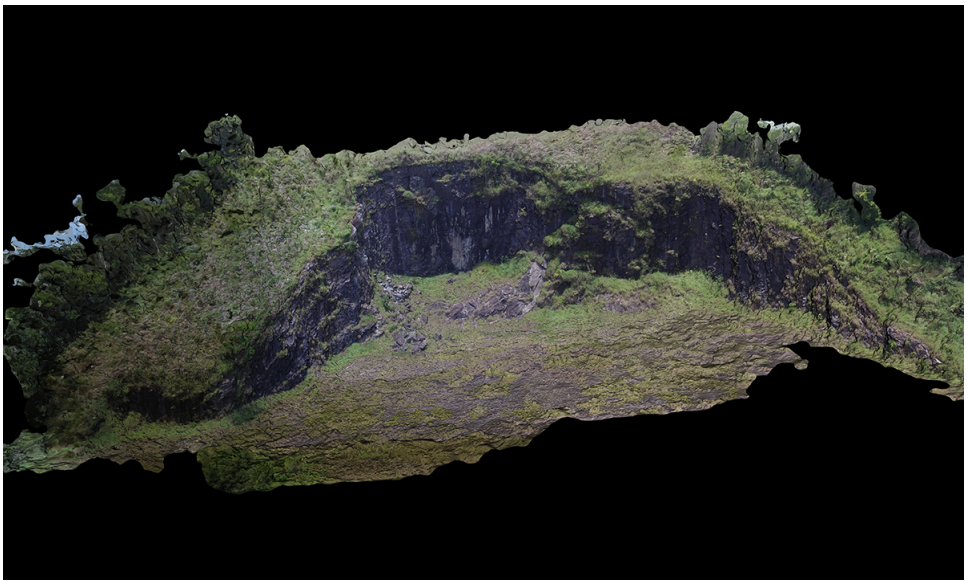


Figure 48 – Reconstructed area.



The second task is a water dam inspection. During this mission, the UAV is in charge of searching for vegetation around the dam. Initially, the mission planning algorithm creates a trajectory for the UAV. The path is initially optimized to take enough images for reconstruction as fast as possible. However, if there is a detection of nonconformities (NCs), the decision-making

system sends a request for path change to Layer 3. Then, after re-planning the mission, the new trajectory data is sent to the low-level control system. Images closer to the inspected area increase the level of details, but also increase the time for the mission. As soon as the NCs are out of the image, the path planning can resume the initial planning. Figure 49 shows the capability of the cognitive architecture of taking actions based on visual image recognition. Initially, the aircraft is acquiring images for 3D reconstruction. But, during the inspection, an NC recognition occurs. As a result, the system requires a path change to approximate the aircraft to the dam. Note that reducing the distance from the inspection target will also increase the number of pixels per square centimeter, which improves the amount of captured details. In the end, it is also possible to note the moment that the last NCs are recognized and the path returns to the previous distance. Figure 50 shows the recognized NCs during the requested path planning change.

Figure 49 – Path changing after recognizing points of interest in the inspection.

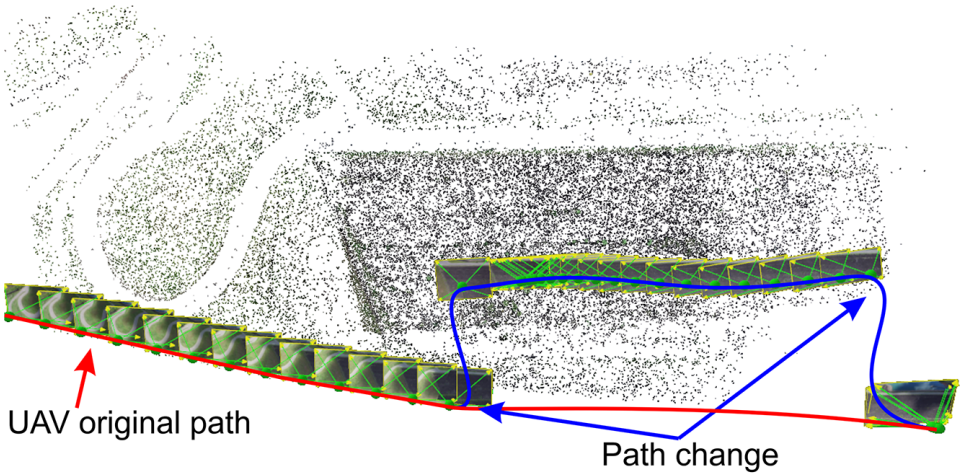


Figure 50 – Recognized points of interest in the image.

Another important feature in the ARCog is the decision making based on flight safety. This experiment shows the aircraft changing its path during the inspection. The UAV suffers

from a wind gust due to its approximation to the dam. Figure 51 illustrates the wind disturbance experienced by the aircraft during the inspection. As a result of this disorder, the motors increased their total power output to withstand the wind force to maintain the position. Figure 52 (b) shows the UAV original path planning and the target position (i.e., dam) measured by the depth sensor in relation to the aircraft body frame. During the wind disturbance, the system requests for a path planning and the aircraft increases its distance from the dam. Figure 52 (a) shows the change in throttle when this event occurred. To avoid the possibility of the aircraft being overwhelmed by the wind, two situations are constantly monitored: (1) the rate of change in throttle; and (2) the maximum throttle. If any of those exceed the acceptable limits, the system requests a new path planning.

Figure 51 – Aircraft suffering a wind disturbance during the inspection.

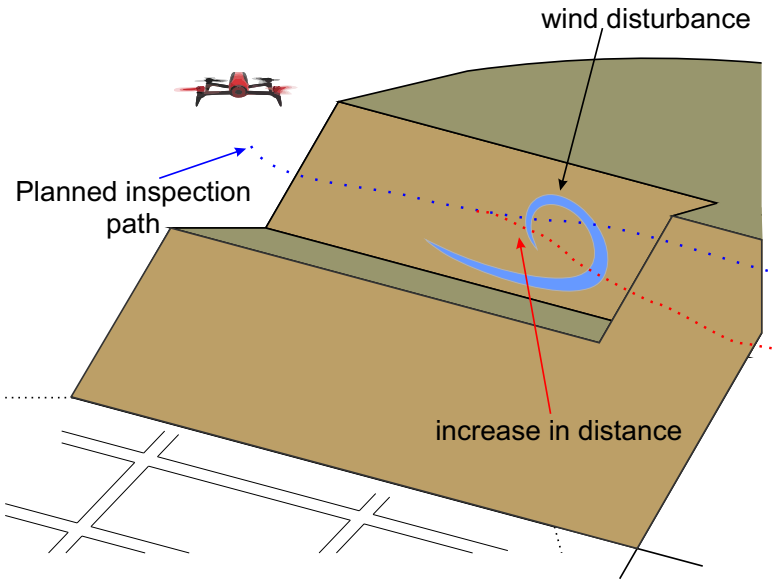
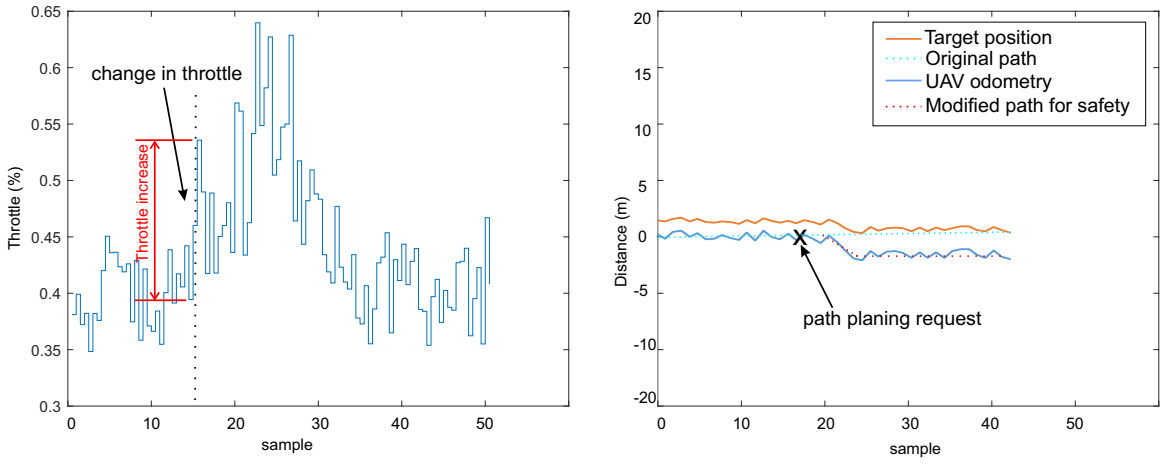


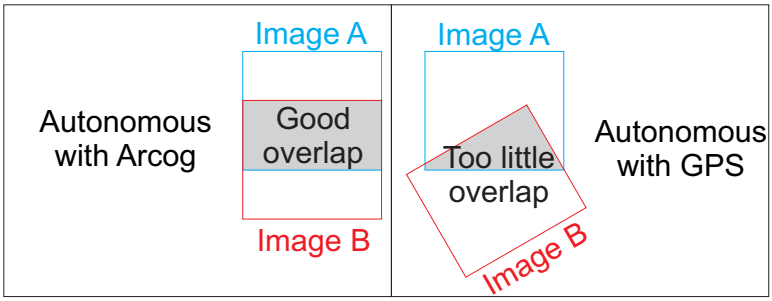
Figure 52 – Modifed UAV path based on safety. (a) UAV's motors increasing their power to maintain the position. (b) Path changing during the wind disturbance.



An interesting way of measuring the capacity provided by the architecture is comparing the results between the autonomous operation and the manual inspection. A great benefit of

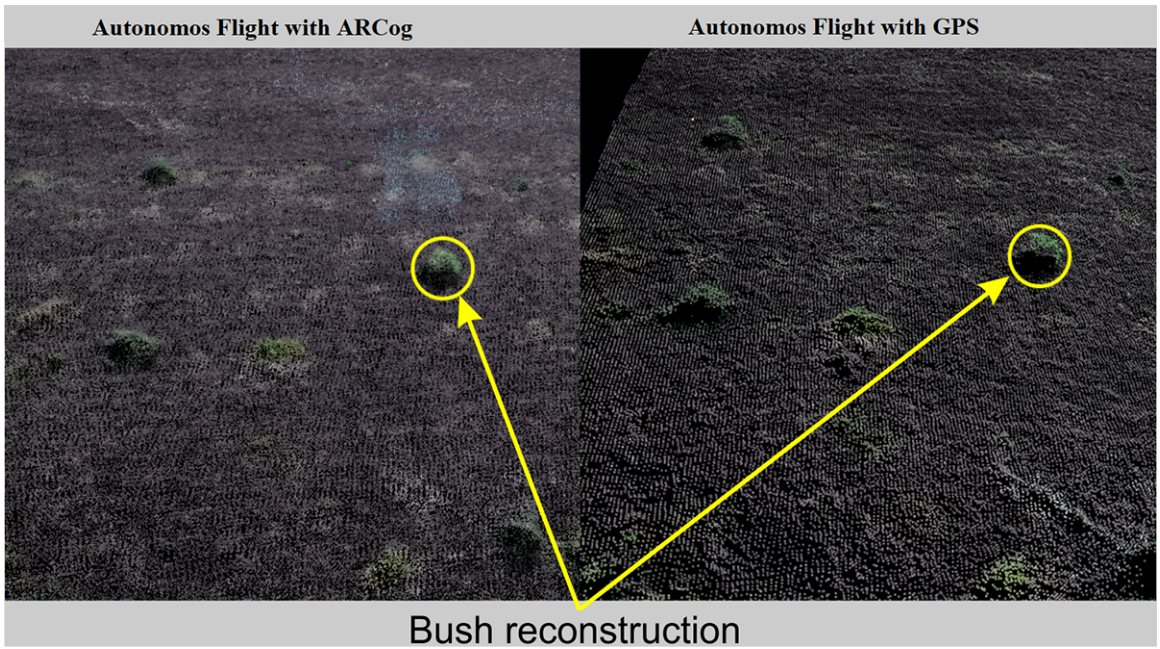
autonomous operation is the ability to capture data with a required amount of overlap and with adequate matching positions and orientations. Figure 53 shows a representation of these image acquisition errors. The autonomous image acquisition can also rely on non-specialized inspectors to produce consistent and accurate results.

Figure 53 – Characteristics of autonomous inspection flight with ARCog and with GPS.



The authors performed a flight over the area using a predefined distance from the target capturing images along with this path to compare the results between the autonomous operation using ARCog and the autonomous using GPS. Figure 54 shows the reconstruction of an area stretch from both methods. The quantitative results are in Table 19. Note that the symbol σ is the standard deviation.

Figure 54 – Quality of autonomous mission using the ARCog and GPS inspection.



5.5 FOG COMPUTING FRAMEWORK

This part of this proposed research presents the results and the respective analyses about the developed model efficacy. The model was only evaluated in simulation, and this part of the work is still in progress. Some numerical assumptions about the problem were presented in

Table 19 – Quantitative comparison between autonomous and manual inspections.

Parameter	Autonomous with ARCog	Autonomous with GPS
Resolution	2 cm	4.47 cm
Mean Error X(σ)	0.07 m	0.51 m
Mean GPS Distance(σ)	0.229 m	0.821 m
Keypoints / image	17701	15495

the previous chapter to show how the model works without defining a specific scenario for the problem. The first parameter is the different levels of data traffic generated by a given number of UAVs. The second one is the characteristic of some mobile network standards. Lastly, the third parameter simulates the levels of processing capability at the fog level.

These parameters illustrate the behavior of the objective functions. For example, we can analyze the data traffic/throughput rate versus fog-cloud computing viability. A question to ask is how much rate among data traffic and throughput must increase before fog computing becomes viable. These analyses should assist the decisions boundaries understanding for each objective function presented in 4.2, 4.3 and 4.7.

5.5.1 EXPERIMENTAL DESIGN OF THE FOG FRAMEWORK

An environment was deployed on the software MATLAB version R2016a [116] to simulate the proposed model. In this experimentation, a broad range of common requirements is selected for the aircraft application and network infrastructure. Moreover, few assumptions about the data are required due to the dependency of a scenario to apply the model. Then, the experimentation consists in selecting a given configuration (i.e., packet processing capability at the fog, network throughput, and arrival rate) and after that, the algorithm works finding solutions for a set of different work distributions between fog and cloud devices.

Missions using UAVs require different types of data. The image is one of the most demanding ones. Thus, it is necessary a bandwidth with a capacity to transmit images with different qualities during the tasks. The bandwidth can be used as a parameter to analyze the throughput requirement for the system operation. In this work, the simulations use bandwidth related to video transmission varying from a single aircraft in 360p up to a team of 6 UAVs transmitting video in 1080p video in 1080p ([117], [118]) as can be seen in Table 20.

Table 20 – Bitrate required for common image sizes used in UAV applications.

Resolution	**Minimum bitrate**	**Maximum bitrate**
360p	400 Kbps	1.000 Kbps
480p	500 Kbps	2.000 Kbps
720p	1.500 Kbps	4.000 Kbps
1080p	3.000 Kbps	6.000 Kbps

The throughput and inherent latency of the network depend mostly on the network

protocol. In this sense, the throughput and latency are studied selecting different types of networks ranging from Edge to 4G, as shown in Table 21 [56]- [58].

Table 21 – Network characteristics of common mobile standards.

Network	**Latency**	**Throughput(Uplink)**
GSM	600-750 ms	40 Kbps
UTMS	500-750 ms	384 Kbps
HSPA	150-400 ms	5.76 Kbps
HSPA+	100-200 ms	11.5 Kbps

The fog processing capability depends on both architecture and processor. It is essential to determine the maximum of load that the fog computing can handle before becomes unstable or present a high processing latency. In this way, the utilization of fog computing in situations where the system cannot process enough packets may degrade the average latency. In this case, the application turns impracticable. Thus, in this work, the processing capability changes from 25% to 100% of incoming packets.

The efficiency of the processor is the most difficult parameter to determine due to its dependence on many factors, such as thermal processor efficiency, operational system and the services provided by the UAV coordinator. This work considers the parameters of the processor matching an x86 processor model z3775g when changing from idle to full utilization. The power consumption was measured using the operational system Ubuntu 14 with LXDE. During the measurements, the system performed image processing, data storage, and generic data processing.

There are many requirements for the architecture implementation. For instance, UAVs intercommunication is a critical issue. However, it will not be a subject of this work. The communication is considered as a high-speed local connection with low power [110] and should ensure that this data exchange has a minimal delay and it does not significantly affect the proposed problem. The information offloaded between fog and cloud devices is provided by a different interface that uses mobile telecommunication, e.g., GPRS networks.

5.5.2 FOG COMPUTING FRAMEWORK RESULTS

Figure 55 shows the first result. The red and blue arrows help in visualizing the increase direction of fog processing rate and throughput, respectively. The parameters were assigned with colors that represent their original configurations. In this figure, the throughput and processing rate are represented respectively by blue and red colors. Note that the values of throughput and processing rate are constant for a set of points with the same color. The throughput affects the latency baseline, as indicated by the blue arrow. The latency behavior in respect with workload distribution depends on the fog processing rate, i.e., if the fog processing is at a particular minimum value, the latency will decrease as the workload is added to the fog level.

Figure 55 – Optimization for packets arrival rate at fog device.

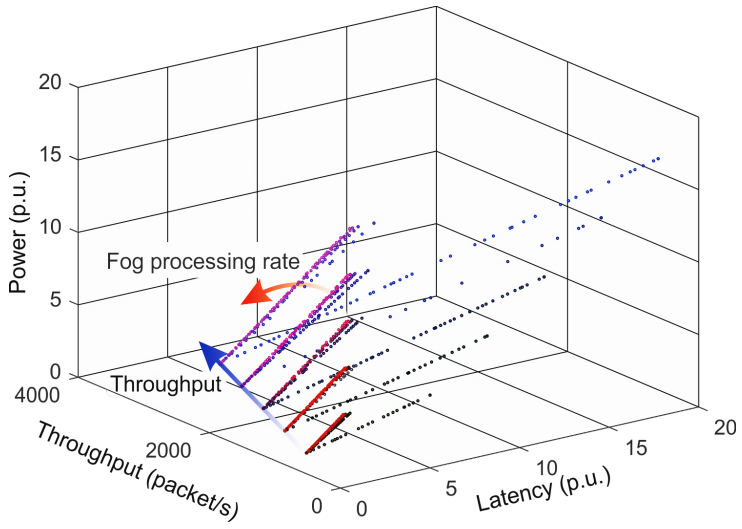
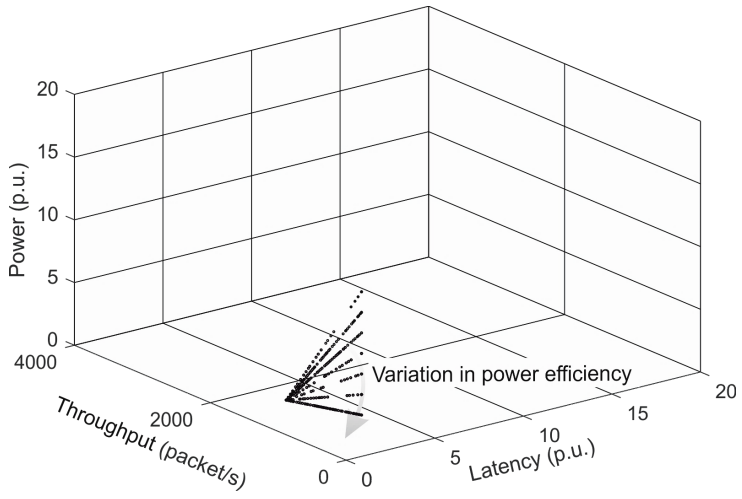


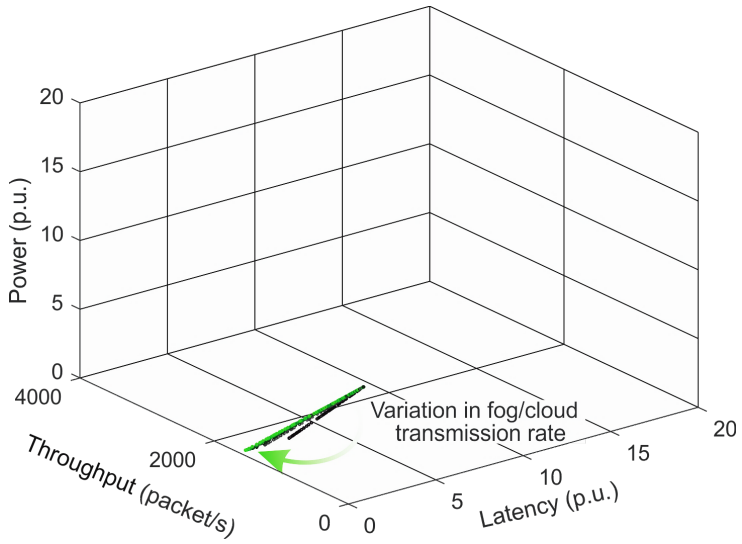
Figure 56 shows the energy efficiency and throughput variation. Note that both parameters present a simple exchange. This is related to the problem design that considers only throughput between fog and cloud computing and power consumption at the fog computing.

Figure 56 – Energy efficiency behavior.



The variation of the transmission rate between fog and cloud computing is shown in Figure 57. This variation analyzes the system behavior as the performance of the communication structure improves. The lines in Figure 57 turns around a point with 100% of workload in the fog device. However, in Figure 55, the curves turns around 100% of workload in the cloud. This indicates that the transmission rate behaves concurrently with the fog processing rate presented in Figure 56. The latency will be improved as more work is transferred to the cloud.

Figure 57 – Optimization of fog and cloud transmission rate.



5.5.2.1 RESULTS FOR MODIFICATION 1

As mentioned before, the first modification in the proposed model considers the power required to offload data from fog to the cloud. The throughput and latency behaviors for a variable number of incoming packets are shown in Figure 58(a). Figure 59(a) exhibits the energy, throughput and latency for a different number of incoming packets and fog processing rate. For comparison purposes, Figure 58(b) and Figure 59(b) exhibit the model behavior regardless the power required for data transmission. The results of Figure 58 do not show significant changes. Hence, the relationship between latency and throughput is not affected by this parameter. However, in Figure 59, this parameter changes the minimum value of energy consumption for a determined configuration.

Figure 58 – Throughput and latency for a variable number of incoming packets. (a) Original model. (b) Modification 1.

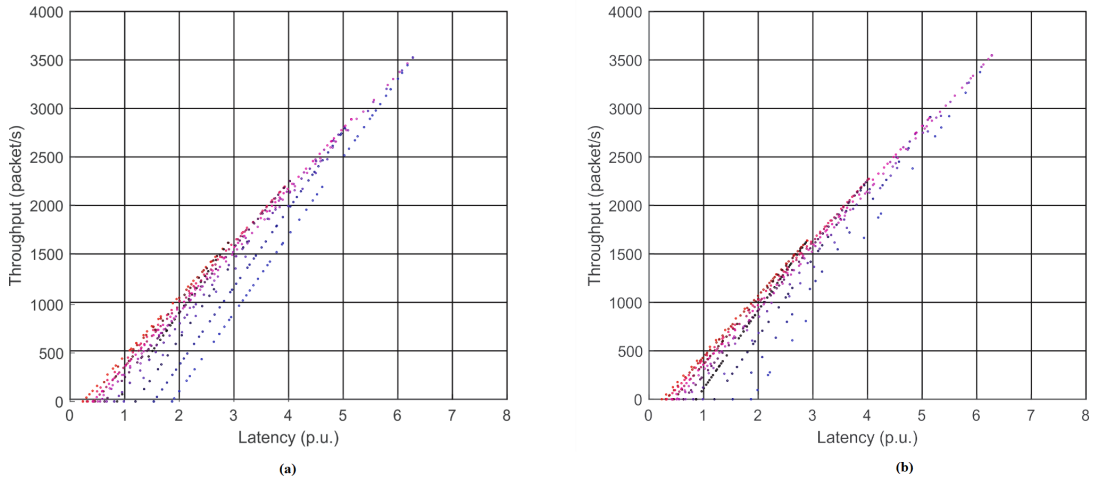
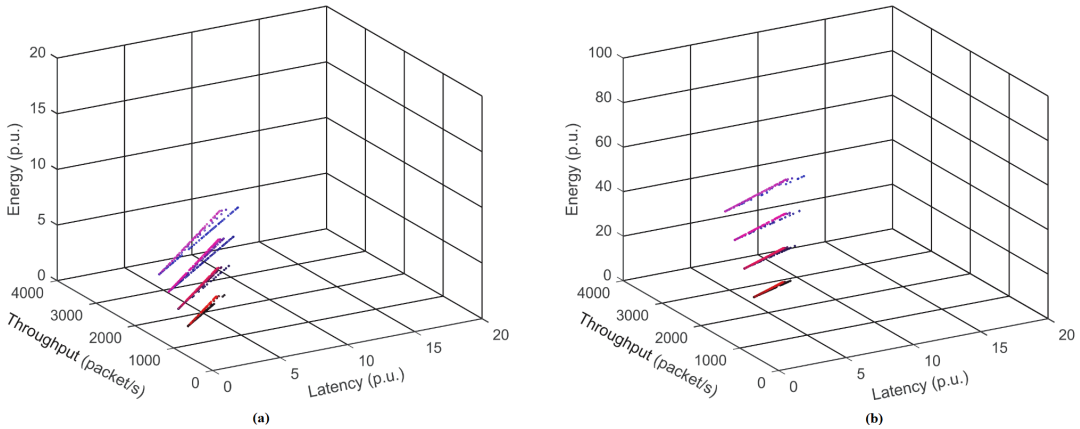


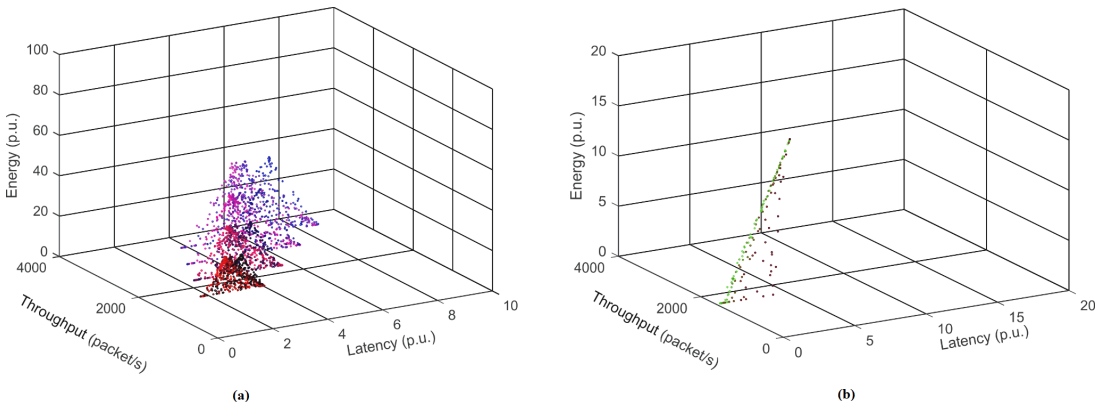
Figure 59 – Energy, throughput and latency for a variable number of incoming packets and fog processing rate. (a) Original model. (b) Modification 1.



5.5.2.2 RESULTS FOR MODIFICATION 2

The second model modification illustrates the effect of randomness in the network on the system operation. We can see the behavior of the parameter $f(d_i^{cloud}, t_i^{(fog-cloud)})$ from 4.9 with a Gaussian distribution and standard deviation of 0.15 over the defined mean value. The colors applied for each parameter are the same from the original model. Figure 60 (a) and Figure 60 (b) show energy, throughput, and latency for a variable number of incoming packets and fog processing rate and a variable fog/cloud transmission rate, respectively. It is possible to note the few changes in the generic behavior of the optimized parameters when compared with Figure 55 and Figure 57.

Figure 60 – Energy, throughput and latency for: (a) a variable number of incoming packets and fog processing rate; and (b) a variable fog-cloud transmission rate.



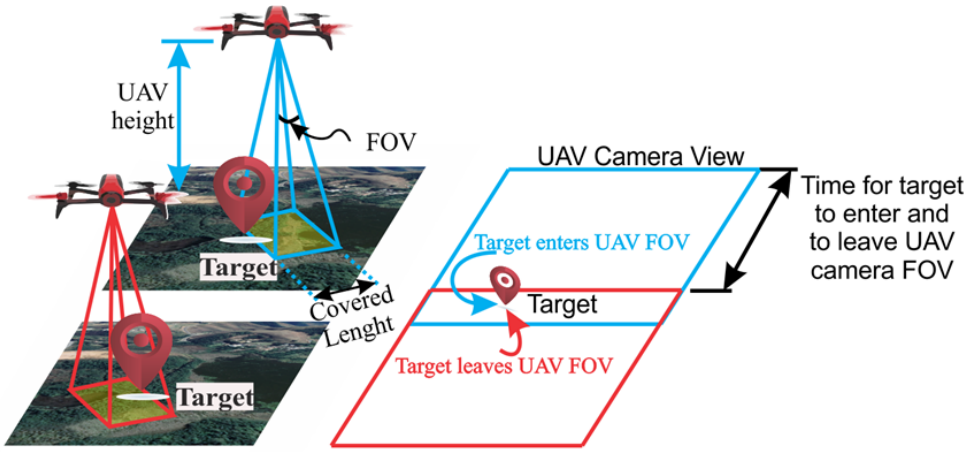
5.5.2.3 MAXIMUM LATENCY ANALYSIS

The feasibility of a fog node addition is tightly connected to the maximum latency allowed by the specific application. However, an analysis of response time and throughput requirements for a generic UAV application is not possible, which is due to the diverse nature

of services and tasks that will impact those requirements. In this sense, a specific scenario is presented to at least indicate how latency requirements can be perceived.

For a SAR environment or inspection in a large area, the main goal is to find a specific target. In this way, if the image data is being processed at the cloud, the processing time should not exceed the sum of time that the prominent target enters the camera FOV and leave it. Thus, in case that the UAV is mapping an area in a specific speed, the decision to track the target should be fast enough to assure that the target is still in the camera FOV during the decision making. Figure 61 presents this process.

Figure 61 – Representation of the target entering and leaving the camera FOV.



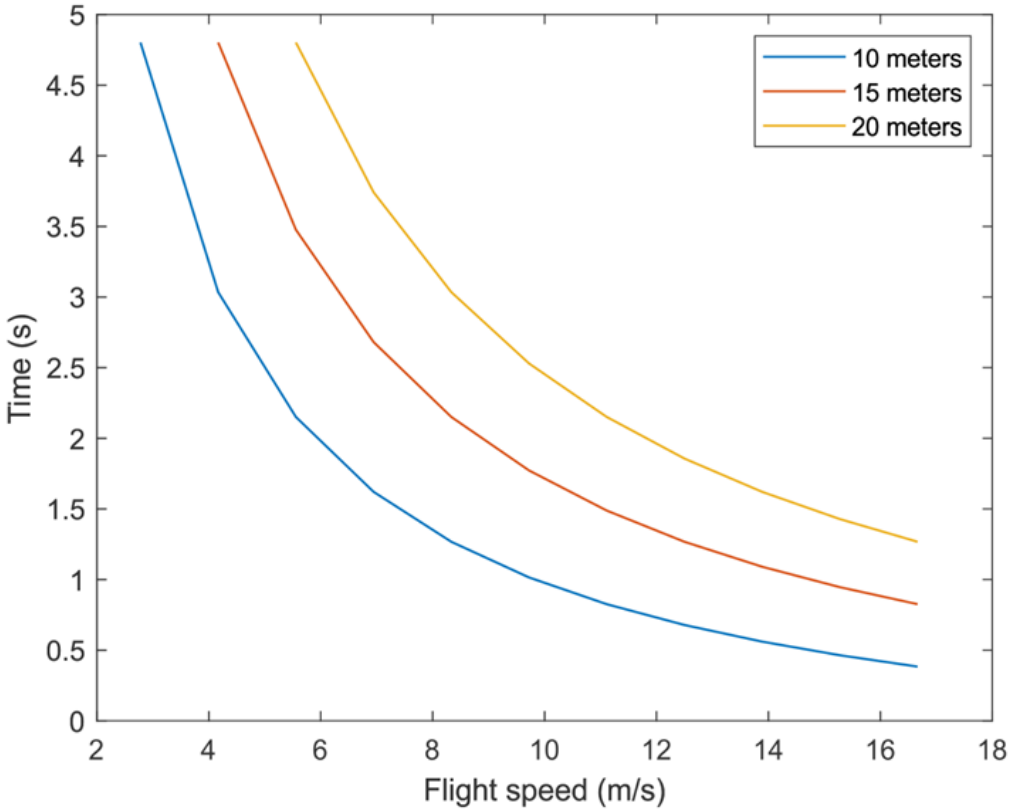
Considering a camera with 94 degrees of FOV and 3:2 of aspect ratio (e.g. Phantom 3 camera), then, it is possible to determine the length covered at a certain height (h) performed by the camera using Equation 5.2 [42].

$$lenght = 2 \cdot h \cdot \tan\left(\frac{\tan^{-1}\left(\frac{3 \cdot \tan(FOV)}{2}\right)}{2}\right) \tag{5.2}$$

Then, it is possible to compute the available time for data processing in the established conditions. Figure 62 shows the results for 10, 15 and 20 meters of height considering the total time to capture and to process the data on the UAV as well as to perform the round trip to the network and to process in the fog/cloud computing. The work presented in [119] suggests that a two-hop latency (A2A-A2G) for sensor data can reach 0.84 seconds. Thus, a round trip would have at least 1.68 seconds, which turn some of the UAV cloud applications unfeasible. Despite being strongly correlated with the proposed scenario, these results showed that the deployment of a fog node can benefit the UAV applications in situations where cloud deployment is not feasible.

The proposed feasibility model is verified through the comparison between the results of Figure 62 and the ones presented in the previous section. The model should allow the designer to

Figure 62 – Total time between a target entering and leaving the UAV camera FOV for determined flight heights.



analyse the feasibility of the fog application according to the changes in the model variables. An example would be the analysis of which flight velocity of Figure 61 would make indispensable in a fog computing application.

5.5.2.4 POWER CONSUMPTION IMPACT ANALYSIS

An important analysis in the feasibility evaluation of applying the proposed architecture is the possible impact in energy consumption. Thus, this subsection presents an analysis concerning a study case on the topic. Considering the parameters of Table 22 for an electric motor X2212 1250kV manufacture by Sunnysky [120] and a standard propeller, it is possible to estimate the energy consumption required to lift a quad-motor rotary wing and a bi-motor fixed-wing aircraft from 500 grams up to 3000 grams using the models in [121] and [122]. This range of weights was chosen due to the fact that they cover most of the commercial UAVs. Note that the motor was chosen to lift the heaviest aircraft. The same configuration was applied to all weights to simplify the assumptions and comparisons. This does not intend to represent the best design choice or system physical constraints.

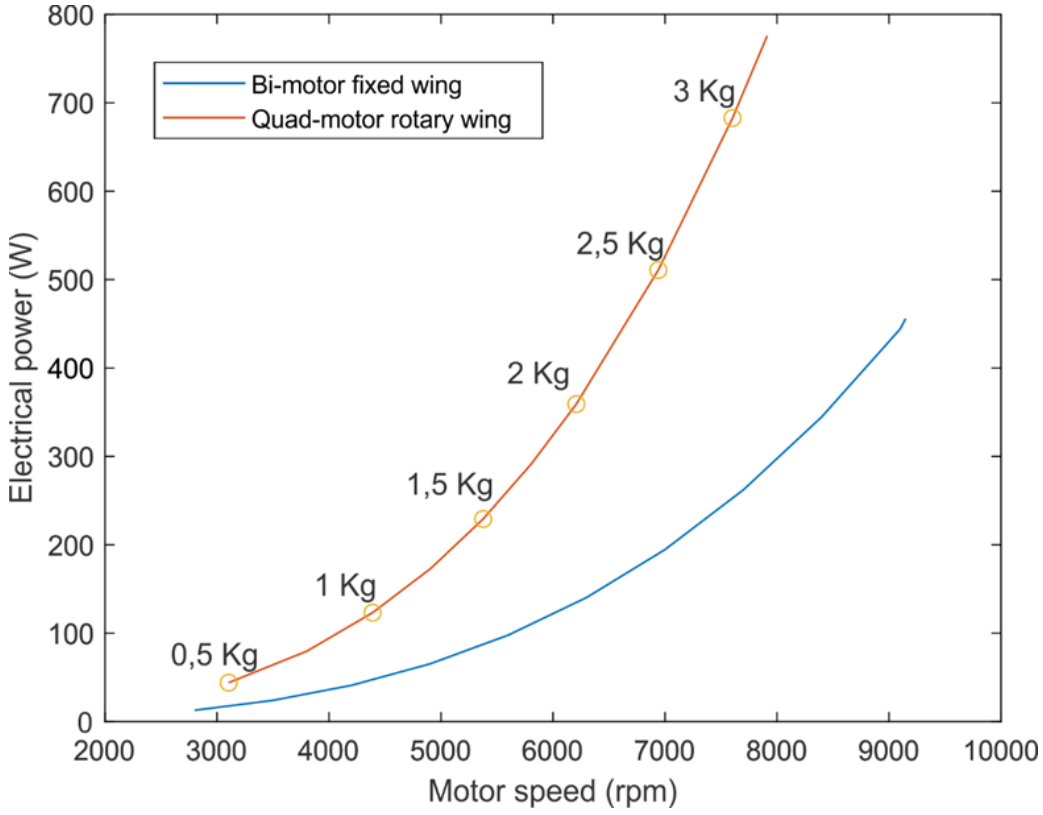
Figure 63 presents the relationship between electrical power and the motor speed for quad-motor rotary wing and a bi-motor fixed-wing aircraft. In the quad-motor curve, this figure presents the minimum power required to hover, i.e., to keep the quadrotor flying. This power was

Table 22 – Electric motor parameter.

Parameter	Autonomous with GPS
Motor Speed	1250 rpm/V
Current Without Load	0.6 A
Motor Resistance	0.079 *omega*
Maximum Power	390 W
Propeller	Diameter: 254mm Pitch: 119 mm
Motor Efficiency	75 - 85%

computed by the amount of thrust required to overcome gravity for the specific weights shown. The same calculations for the minimum speed/power for a fixed wing are more complex and they largely depend on the aerodynamic aircraft design. As a reference, a 72 dm^2 fixed-wing aircraft with drag coefficient of 0.05 at standard temperature and pressure conditions will require between 2400 and 5400 rpm to maintain leveled flight for weight values between 0.5Kg and 3Kg. This ultimately results in powers requirements between 13 W and 65 W accordingly to Figure 63.

Figure 63 – Relationship between motor speed and power consumption for rotary and fixed-wing aircraft.



The energy consumption impact of adding an onboard computer into a UAV is analyzed through this data considering the weight impact from a single board computer and a dedicated battery bank. The power required in the analysis is simplified assuming that the onboard computer has a dedicated isolated battery bank.

For a rotary wing (adding 250 grams), the power required to sustain a flight increases

from 9W in an aircraft with 0.5 kg grams to 23.3 W in the 3 kg model. For the fixed-wing aircraft, this analysis is more complex and the additional power required to sustain a flight increases around 3 W for the same mentioned design. Note that the added weight significantly impacts the power needed to maintain the flight, which consequently affects energy consumption and aircraft flight time. In this sense, the onboard computer has to be efficient to require lightweight batteries and to reduce the mentioned impact as much as possible. It is also relevant to note that this result is in accordance with other research works in this field. For instance, [123] suggests that the relationship between information processing and communication to the motors consumption is 20/80.

5.6 PARTIAL CONCLUSIONS

This chapter presented the results of a cognitive architecture for UAVs namely ARCog. These results aimed at showing the robustness of the proposed architecture for recognizing activities in surveillance, search and rescue and inspection missions. The architecture formalization includes five main blocks. These main components guarantee the execution of the architecture for autonomous missions. The experiments confirm the successful use of the architecture to be applied in semi-autonomous missions, where the UAV presented some degree of autonomy in the decision-making process when making use of the proposed data flow along with Case-Based Reasoning.

This chapter also showed the EKF methodology to predict the moving target trajectory and UAV movement without human intervention. It was fused data from optical flow calculated using an OpenCV function for background removal with the information from IMU and GPS to accomplish this task. This work showed the possibility to integrate multi-sensorial information to compute object inertial coordinates in real-time with a respectful accuracy. The results obtained in the simulation showed improvements that are in the expectation range within the cited literature. During the execution, the CPU usage remained lower than 3% to process the Kalman algorithm, showing the possibility to compute the algorithm in real-time with a relative low computation power.

The UAV sensors obtain information from the scenes. The CBR algorithm enables online learning of the behaviors containing in these scenes by inferring over the incoming data and past solutions to decide what actions the UAV should take based on what is happening. Experimental results showed that the architecture is efficient and can respond to changes in scenarios. The ARCog exhibits flexibility to interpret complex situations, aiding operators in the detection of unsafe conditions.

6 FINAL CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

Robotic systems require a combination of several specific and interrelated building blocks to reach autonomous operation capabilities. Many open-source architectures for UAVs have been developed to guarantee a semi-autonomous operation. However, in most of the cases, the autonomy level is limited and restricted by the application. Therefore, this research has described an intelligent software architecture for UAVs with an optimized definition of the blocks. The architecture lays the groundwork for the development of a software platform aligned with the current requirements of state-of-the-art technologies. The proposed architecture is designed to have the capability of high-level decision making, such as understanding, interpretation and goal-oriented reasoning. The main contribution of this research work is the capability of supporting the latest advances in technologies and the development of an organized paradigm to support decision making and high-level cognition. Besides, this work describes the justification of each functional block and its organization within the architecture.

The architecture formalization includes five main blocks divided in four layers to guarantee the architecture autonomous execution. Each module solves some specified tasks. Moreover, these modules encapsulate a certain level of abstraction. Note that the representation of the signals that the functional block uses to solve a given task is more abstract as the level is becoming higher. An important contribution of this research is the development of an organized paradigm to support decision making and high-level cognition. Besides, this research describes the justification of each functional block and its organization within the architecture.

The ARCog works in the following way. After sensing the information in the lower level, the modified automaton organizes and processes the sensory information to build relations. This innovative approach propagates the certainty of each recognized instance associated with the automata, and the probabilities can be mapped by weights or normalization of the data. After this process, the CBR algorithm receives the structured information to perform the inference. Besides, it displays the analysis of the scene to the user. The experiments showed that the proposed cognitive architecture was able to update the automata effectively given a sequence of information. Furthermore, the architecture was flexible enough to interpret complex situations to assist the human operators.

The experiments were divided among algorithms and applications. The results confirmed the success of ARCog in semi-autonomous missions, such as Inspection, Search and Rescue and Surveillance. The UAV presented some degree of autonomy in the decision process when using the proposed data flow along with CBR methodology. The embedded hardware is the main limitation of the architecture. The hardware may have specific requirements for additional software and device, processing capability and memory. The major challenge of this work was the complete implementation of the proposed architectures in a real embedded system with specific

operating requirements from scratch. For this, all the algorithms were tested separately through simulations and particular tests. Another significant challenge was the practical development of all levels of architecture and their communications for different missions.

This thesis also introduced a model to investigate the fog-cloud computing cooperation to overcome the throughput and latency involving multiple aircraft in areas with low communication infrastructure. Besides, the analyzed model includes the power consumption limitations of UAVs in the proposed framework. The model assigns a UAV as head coordinator in a fog computing level to process and to control the communications between the nearest nodes and the cloud. Thus, this coordinator manages the computational offloading between cloud and fog. Besides, the head coordinator enables the continuity of the autonomous operation even when it is not connected to the ground station. This process results in a lower latency when dealing with the real-time data provided by the aircraft. The proposed model differs from previous works by capturing performance behavior not totally explored in UAV scenarios including the limited amount of energy, processing capability restrictions at the fog computing and limited throughput among fog and cloud computing. Those characteristics may also be found in other applications (e.g., IoT) which makes the model even more interesting. Note that the implementation of a FOG node is a work in progress. However, the simulations results shown ots potential application.

An extensive evaluation of the fog-cloud model was reported. The simulations and numerical results showed that the model can be used for fog-cloud computing evaluation when considering latency, energy and bandwidth constraints. The model covers three experimental scenarios with an increasing degree of accuracy with respect to the real world. The architecture layout demonstrated the possibility to overcome the throughput and latency limitations involving multiple aircraft during missions in restricted areas when applying fog-cloud computing cooperation. The contributions of this research can help researchers understand and design UAVs to further assist missions.

6.2 FUTURE WORK

Few extensions are foreseen in this research work. First, the fog-cloud will be deployed using real UAVs to validate the model further. Therefore, it is also intended to investigate the impact of other factors, such as battery time, number of head coordinators and type of data processing. Moreover, the comparison with other applications models such as IoT is proposed to show different perspectives for this research work.

This work opens the possibility of several future developments, such as the uncertainty-based decision support with fuzzy or Bayesian inferences process, the use of online reinforcement learning to train new behaviors during the mission, the application of context-aware methodologies to increase the reasoning in the decision-making process. These future developments in the algorithms may bring more satisfactory results

In terms of evaluation, albeit that the proposed ARCog architecture considers global missions, our assessment was focused on a semi-autonomous outdoor search and rescue mission. Thus, additional experiments are intended to include more complex scenarios and different mission goals. These will evaluate the architecture capabilities for an extended range of situations, including multiple aircraft cooperation, hazardous environments, and UAV failure.

7 AUTHOR'S SCIENTIFIC PRODUCTION

International Conference Papers:

- ALMEIDA, C. C.; ALMEIDA, P.S.; MONTEIRO, N. R. C.; PINTO, M. F. BRAGA, H. A. C. LED - based Electronic System to Support Plant Physiology Experiments. IEEE 23rd International Symposium on Industrial Electronics (ISIE). Turkey, 2014.
- ALMEIDA, C. C.; ALMEIDA, P.S.; PINTO, M. F.; VALLE, R.L.; MARTINS, H.N.; BRAGA, H.A.C. A Fast Dynamics and PWM-Dimmable LED Driver for Accurate Control of Illumination in Plants Physiology Experiments. IEEE/IAS International Conference on Industry Applications (INDUSCON). Brazil, 2014.
- PINTO, M. F.; MENDONÇA, T. R. F; OLIVI, L. R.; COSTA, E. B. Costa; MARCATO, A.L.M. A modified approach of potential field method for control of trajectory tracking and obstacle avoidance. IEEE/IAS International Conference on Industry Applications (INDUSCON). Brazil, 2014.
- MENDONÇA, T. R. F; PINTO, M. F.; DUQUE, C.A. Least Squares Optimization of Zero Crossing Technique for Frequency Estimation of Power System Grid Distorted Sinusoidal Signals. IEEE/IAS International Conference on Industry Applications (INDUSCON). Brazil, 2014.
- PINTO, M. F.; SOARES, G. M.; MENDONÇA, T. R. F.; ALMEIDA, P.S.; BRAGA, H. A. C. Smart Modules for Lighting System Applications and Power Quality Measurements. IEEE/IAS International Conference on Industry Applications (INDUSCON). Brazil, 2014.
- MENDONÇA, T. R. F.; PINTO, M.F.; DUQUE, C. Variable Window Length Applied to A Modified Hanning Filter For Optimal Amplitude Estimation of Power Systems Signals. 2015 IEEE PES generic Meeting. EUA, 2015.
- PINTO, M. F.; MENDONÇA, T. R. F.; DUQUE, C.A.; BRAGA, H. A. C. Power Quality Measurements Embedded in Smart Lighting Systems. IEEE 24th International Symposium on Industrial Electronics (ISIE). Brazil, 2015.
- PINTO, M. F.; MENDONÇA, T. R. F.; COELHO, F.; BRAGA, H. A. C. Economic Analysis of a Controllable Device with Smart Grid Features Applied to LED Street Lighting System. IEEE 24th International Symposium on Industrial Electronics (ISIE). Brazil, 2015.
- MENDONÇA, T. R. F.; PINTO, M.F.; MARCATO, A. L. M. Electric Field Intensity for Nonlinear Classifier: A Novel Approach. IEEE 24th International Symposium on Industrial Electronics (ISIE). Brazil, 2015.

- MENDONÇA, T. R. F.; PINTO, M.F.; DUQUE, C. A. Adjustable Window for Parameter Estimation Considering the Time-Varying Frequency of Power Systems Signals. IEEE 24th International Symposium on Industrial Electronics (ISIE). Brazil, 2015.
- PINTO, M.F.; MENDONCA, T.R.F.; DUQUE, C.A.; BRAGA, H.A.C. Street Lighting System for Power Quality Monitoring and Energy-Efficient Illumination Control. In: IEEE International Symposium on Industrial Electronics. EUA. 2016.
- PINTO, M.F.; MELO, A. G.; MARCATO, A.L. M.; URDIALES, C. Case-Based Reasoning Approach Applied to Surveillance System Using an Autonomous Unmanned Aerial Vehicle. 26th IEEE International Symposium on Industrial Electronics. Scotland. 2017.
- MENDONCA, T.R.F.; COLLINS, M.E.; PINTO, M.F.; GREEN, T.C. Decentralization of Power Flow Solution for Facilitating Active Network Management. 24th International Conference on Electricity Distribution. Scotland. 2017.
- COELHO, F.O.; SOUZA, J.P.C.; PINTO, M.F.; MARCATO, A.L.M. Direct-DRRT*: A RRT Improvement Proposal. CONTROLO. Portugal. 2018.
- PINTO, M.F.; COELHO, F.O.; SOUZA, J.P.C.; MELO, A. G.; MARCATO, A.L.M.; URDIALES, C. E KF Design for Online Trajectory Prediction of a Moving Object Detected Onboard of a UAV. CONTROLO. Portugal. 2018.

Brazilian Conference Papers:

- PINTO, M. F.; MELO, A. G.; LOPES, L. C. G.; JUNIOR, L. O. A. Calibração De Imagens Estereoscópicas De Objetos Em 3 Dimensões Para Geração Automática De Trajetória De Manipuladores Robóticos Industriais. X Simpósio Brasileiro de Automação Inteligente (SBAI). 2011.
- PINTO, M. F.; LANES, M. M.; TOLEDO, O.M. Módulo Didático Flexível de Baixo Custo para Sensoriamento de Velocidade. Congresso Brasileiro de Automática (CBA). 2012.
- PAULA, B. F. S.; PINHEIRO, C. V.; PINTO, M. F.; SILVEIRA, D. D. Comparação estatística entre a educação presencial e a educação a distância. XLII Congresso brasileiro de educação em engenharia (COBENGE). 2014.
- PINTO, M. F.; MELO, A. G.;LANES, M. M.; TOLEDO, O.M. Desenvolvimento de um kit didático para controle e acionamento de motores CC. XLII Congresso brasileiro de educação em engenharia (COBENGE). 2014
- COELHO, F.O.; SOUZA, J.P.C.; PINTO, M.F.; MACIEL, G.M.; MARCATO, A.L.M. Filtro de Kalman Estendido Baseado em Visão Computacional Aplicado à Localização e Sequestro de Robôs Móveis. XIII Simpósio Brasileiro de Automação Inteligente (SBAI). 2017.

- PINTO, M.F.; MELO, A. G.; URDIALES, C; MARCATO, A.L. M. Remoção Dinâmica de Plano de Fundo em Imagens Aéreas em Movimento. XXII Congresso Brasileiro de Automática (CBA). 2018.

Journal:

- PINTO, M.F.; MARCATO, A. L. M.; HONÓRIO, L. M.; MELO, A. G.; URDIALES, C. A Framework for Analyzing Fog-Cloud Computing Cooperation Applied to Information Processing of UAVs. Wireless Communications and Mobile Computing. Volume 2019, 1-14, 2019.
- NETO, W. A.; PINTO, M.F.; MARCATO, A. L. M.; SILVA Jr, I. C.; FERNANDES, D. A. Mobile Robot Localization Based on the Novel Leader-Based BatAlgorithm. Journal of Control, Automation and Electrical Systems, 2019. https://doi.org/10.1007/s40313-019-00453-2

REFERENCES

[1] M. Seleckỳ, M. Rollo, P. Losiewicz, J. Reade, and N. Maida, "Framework for incremental development of complex unmanned aircraft systems," in *Integrated Communication, Navigation, and Surveillance Conference (ICNS), 2015*. IEEE, 2015, pp. J3–1.

[2] Y. Du, C. W. de Silva, and D. Liu, "A multi-agent hybrid cognitive architecture with self-awareness for homecare robot," in *Computer Science & Education (ICCSE), 2014 9th International Conference on*. IEEE, 2014, pp. 223–228.

[3] L. Tao, "Development of dam safety management system," *Dam and Safety*, 2011.

[4] S. N. Jonkman, J. K. Vrijling, and A. C. W. M. Vrouwenvelder, "Methods for the estimation of loss of life due to floods: a literature review and a proposal for a new method," *Natural Hazards*, vol. 46, no. 3, pp. 353–389, 2008.

[5] D. S. Bowles, "Estimating life loss for dam safety risk assessment-a review and new approach," in *USENIX Security Symposium*, 2004.

[6] D. González-Aguilera, J. Gómez-Lahoz, and J. Sánchez, "A new approach for structural monitoring of large dams with a three-dimensional laser scanner." *Sensors*, vol. 8, no. 9, pp. 5866–5883, 2008.

[7] A. Khaloo, D. Lattanzi, A. Jachimowicz, and C. Devaney, "Utilizing uav and 3d computer vision for visual inspection of a large gravity dam," *Frontiers in Built Environment*, vol. 4, 2018.

[8] G. Buffi, P. Manciola, S. Grassi, M. Barberini, and A. Gambi, "Survey of the ridracoli dam: Uav–based photogrammetry and traditional topographic techniques in the inspection of vertical structures," *Geomatics, Natural Hazards and Risk*, vol. 8, no. 2, pp. 1562–1579, 2017.

[9] T. Özaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Inspection of penstocks and featureless tunnel-like environments using micro uavs," in *Field and Service Robotics: Results of the 9th International Conference*, vol. 105, 2015, pp. 123–136.

[10] L. Snidaro, M. Belluz, and G. L. Foresti, "Modelling and managing domain context for automatic surveillance systems," in *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*. IEEE, 2009, pp. 238–243.

[11] M. Gótzy, D. Hetényi, and L. Blázovics, "Aerial surveillance system with cognitive swarm drones," in *Cybernetics & Informatics (K&I), 2016*. IEEE, 2016, pp. 1–6.

[12] A. Chakrabarty, R. Morris, X. Bouyssounouse, and R. Hunt, "Autonomous indoor object tracking with the parrot ar. drone," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*. IEEE, 2016, pp. 25–30.

[13] K. Boudjit and C. Larbes, "Detection and implementation autonomous target tracking with a quadrotor ar. drone," in *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on*, vol. 2. IEEE, 2015, pp. 223–230.

[14] M. Y. K. Tani, A. Lablack, A. Ghomari, and I. M. Bilasco, "Events detection using a video-surveillance ontology and a rule-based approach," in *European Conference on Computer Vision*. Springer, 2014, pp. 299–308.

[15] S. Satterfield, T. Reichherzer, J. Coffey, and E. El-Sheikh, "Application of structural case-based reasoning to activity recognition in smart home environments," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 1. IEEE, 2012, pp. 1–6.

[16] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang, "Towards automatic power line detection for a uav surveillance system using pulse coupled neural filter and an improved hough transform," *Machine Vision and Applications*, vol. 21, no. 5, pp. 677–686, 2010.

[17] T. Roth-Berghofer, J. A. Recio Garcia, C. Sauer, K. Bach, K.-D. Althoff, B. Diaz-Agudo, and P. A. Gonzales Calero, "Building case-based reasoning applications with mycbr and colibri studio," 2012.

[18] M. P. Schützenberger, "On the definition of a family of automata," *Information and control*, vol. 4, no. 2-3, pp. 245–270, 1961.

[19] D. E. Rumelhart, "Schemata: The building blocks of cognition," in *Theoretical issues in reading comprehension*. Routledge, 2017, pp. 33–58.

[20] T. Sangpanasthada, "The relationship between cognitive development and language learning in a cross-cultural context: A review of the literature and its implication," *Brock Education Journal*, vol. 15, no. 2, 2006.

[21] N. Chomsky and D. W. Lightfoot, *Syntactic structures*. Walter de Gruyter, 2002.

[22] P. Langley, "An adaptive architecture for physical agents," in *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*. IEEE, 2005, pp. 18–25.

[23] Y. Yang, C. Fermuller, and Y. Aloimonos, "A cognitive system for human manipulation action understanding," in *the Second Annual Conference on Advances in Cognitive Systems (ACS)*, vol. 2. Citeseer, 2013.

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

[25] C. Van Dang, T. T. Tran, Y.-B. Shin, K.-J. Gil, D. Ngo, B.-S. Kang, G.-D. Lee, and J.-W. Kim, "Application of a soar-ros agent on finding the correct object," 2017.

[26] J. T. Ball, "Advantages of act-r over prolog for natural language analysis," in *Proceedings of the 22nd Annual Conference on Behavior Representation in Modeling and Simulation*. Citeseer, 2013.

[27] J. E. Laird, K. R. Kinkade, S. Mohan, and J. Z. Xu, "Cognitive robotics using the soar cognitive architecture," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence, Cognitive Robotics*, 2012.

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

[29] J. W. Tweedale, "A review of cognitive decision-making within future mission systems," *Procedia Computer Science*, vol. 35, pp. 1043–1052, 2014.

[30] C. C. Insaurralde, "Service-oriented agent architecture for unmanned air vehicles," in *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd*. IEEE, 2014, pp. 8B1–1.

[31] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. S. Fernández, and P. Campoy, "A multi-layered component-based approach for the development of aerial robotic systems: the aerostack framework," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2-4, pp. 683–709, 2017.

[32] C. Sampedro, H. Bavle, J. L. Sanchez Lopez, R. Suarez Fernandez, A. Rodriguez Ramos, M. Molina, and P. Campoy Cervera, "A flexible and dynamic mission planning architecture for uav swarm coordination," in *Proceedings of 2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. ETSI_Informatica, 2016.

[33] D. Erdos, A. Erdos, and S. E. Watkins, "An experimental uav system for search and rescue challenge," *IEEE Aerospace and Electronic Systems Magazine*, vol. 28, no. 5, pp. 32–37, 2013.

[34] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, "Multiple moving targets surveillance based on a cooperative network for multi-uav," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 82–89, 2018.

[35] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas, "Anytime planning for decentralized multirobot active information gathering," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1025–1032, 2018.

[36] S. Emel'yanov, D. Makarov, A. I. Panov, and K. Yakovlev, "Multilayer cognitive architecture for uav control," *Cognitive Systems Research*, vol. 39, pp. 58–72, 2016.

[37] V. Dias, R. Moreira, W. Meira, and D. Guedes, "Diagnosing performance bottlenecks in massive data parallel programs," in *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*. IEEE, 2016, pp. 273–276.

[38] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018, pp. 103–130.

[39] J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner, "Ehopes: Data-centered fog platform for smart living," in *Telecommunication Networks and Applications Conference (ITNAC), 2015 International*. IEEE, 2015, pp. 308–313.

[40] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug 2016.

[41] 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 Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, Firstquarter 2018.

[42] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[43] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Dec 2014, pp. 325–329.

[44] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. New York, NY, USA: ACM, 2015, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/2757384.2757397

[45] P. K. Sharma, M. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for iot," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[46] V. Mushunuri, A. Kattepur, H. K. Rath, and A. Simha, "Resource optimization in fog enabled iot deployments," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 6–13.

[47] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *2017 IEEE International Conference on Edge Computing (EDGE)*, June 2017, pp. 17–24.

[48] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1123–1152, Secondquarter 2016.

[49] S. Mahmoud and N. Mohamed, "Broker architecture for collaborative uavs cloud computing," in *2015 International Conference on Collaboration Technologies and Systems (CTS)*, June 2015, pp. 212–219.

[50] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "Uavfog: A uav-based fog computing for internet of things," in *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug 2017, pp. 1–8.

[51] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications," *Journal of Network and Computer Applications*, vol. 82, pp. 152 – 165, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517300188

[52] Y. Liu, J. E. Fieldsend, and G. Min, "A framework of fog computing: Architecture, challenges, and optimization," *IEEE Access*, vol. 5, pp. 25 445–25 454, 2017.

[53] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 3909–3914.

[54] T. Wang, J. Zeng, Y. Lai, Y. Cai, H. Tian, Y. Chen, and B. Wang, "Data collection from wsns to the cloud based on mobile fog elements," *Future Generation Computer Systems*, 2017.

[55] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2018.

[56] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and software architecture for fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 44–53, 2017.

[57] S. Kitanov and T. Janevski, "Fog computing as a support for 5g network," *Journal of Emerging research and solutions in ICT*, vol. 1, no. 2, pp. 47–59, 2016.

[58] Y. Sun, T. Dang, and J. Zhou, "User scheduling and cluster formation in fog computing based radio access networks," in *Ubiquitous Wireless Broadband (ICUWB), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–4.

[59] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1728–1739, 2016.

[60] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[61] F. Zeng, Y. Ji, and M. D. Levine, "Contextual bag-of-words for robust visual tracking," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1433–1447, 2018.

[62] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[63] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[64] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[65] M. Vidal-Naquet and S. Ullman, "Object recognition with informative features and linear classification." in *ICCV*, vol. 3, 2003, p. 281.

[66] E. Unlu, E. Zenou, and N. Riviere, "Ordered minimum distance bag-of-words approach for aerial object identification," in *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[67] T. Moranduzzo, A. Zeggada, and F. Melgani, "A fast screening method for detecting cars in uav images over urban areas," in *Urban Remote Sensing Event (JURSE), 2015 Joint*. IEEE, 2015, pp. 1–4.

[68] J. Fortuna, D. Schuurman, and D. Capson, "A comparison of pca and ica for object recognition under varying illumination," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 3. IEEE, 2002, pp. 11–15.

[69] N. Pinto, Y. Barhomi, D. D. Cox, and J. J. DiCarlo, "Comparing state-of-the-art visual features on invariant object recognition tasks," in *Applications of computer vision (WACV), 2011 IEEE workshop on*. IEEE, 2011, pp. 463–470.

[70] L. Juan and O. Gwun, "A comparison of sift, pca-sift and surf," *International Journal of Image Processing (IJIP)*, vol. 3, no. 4, pp. 143–152, 2009.

[71] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[72] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[73] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[74] L. Itti, "Models of bottom-up and top-down visual attention," Ph.D. dissertation, California Institute of Technology, 2000.

[75] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, p. 2129, 2009.

[76] C. F. for Mobile Platforms, "Meeussen, w," http://www.ros.org/reps/rep-0105.html, 2010, accessed in 09/08/2017.

[77] S. U. of Measure and C. Conventions, "Foote, t and purvis, m," http://www.ros.org/reps/rep-0103.html, 2010, accessed in 09/08/2017.

[78] J. Candamo, R. Kasturi, and D. Goldgof, "Using color profiles for street detection in low-altitude uav video," in *Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications VI*, vol. 7307. International Society for Optics and Photonics, 2009, p. 73070O.

[79] A. Qadir, W. Semke, and J. Neubert, "Vision based neuro-fuzzy controller for a two axes gimbal system with small uav," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 1029–1047, 2014.

[80] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*. IEEE, 2005, pp. 65–72.

[81] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[82] K. Pastra and Y. Aloimonos, "The minimalist grammar of action," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 367, no. 1585, pp. 103–117, 2012.

[83] A. Guha, Y. Yang, C. Fermuuller, and Y. Aloimonos, "Minimalist plans for interpreting manipulation actions," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5908–5914.

[84] K. Chatterjee, L. Doyen, and T. A. Henzinger, "Probabilistic weighted automata," in *International Conference on Concurrency Theory*. Springer, 2009, pp. 244–258.

[85] M. Almeida, N. Moreira, and R. Reis, "Testing the equivalence of regular languages," *arXiv preprint arXiv:0907.5058*, 2009.

[86] T. Das, "Intelligent techniques in decision making: A survey," *Indian Journal of Science and Technology*, vol. 9, no. 12, 2016.

[87] S.-H. Liao, "Expert system methodologies and applications—a decade review from 1995 to 2004," *Expert systems with applications*, vol. 28, no. 1, pp. 93–103, 2005.

[88] R. C. Schank, *Dynamic memory: A theory of reminding and learning in computers and people*. cambridge university press Cambridge, 1982, vol. 240.

[89] J. L. Kolodner, "Reconstructive memory: A computer model," *Cognitive science*, vol. 7, no. 4, pp. 281–328, 1983.

[90] E. L. Rissland, J. Kolodner, and D. Waltz, "Case-based reasoning from darba:"machine learning program plan"," *Proceedings of the case-Based ReasoningWorkshop. San Mateo, CA: Morgan Kaufmann*, pp. 1–13, 1998.

[91] T. Ohko, K. Hiraki, and Y. Anzai, "Lemming: A learning system for multi-robot environments," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 2. IEEE, 1993, pp. 1141–1146.

[92] J. M. Peula, C. Urdiales, I. Herrero, and F. Sandoval, "Implicit robot coordination using case-based reasoning behaviors," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5929–5934.

[93] S. V. Shokouhi, P. Skalle, and A. Aamodt, "An overview of case-based reasoning applications in drilling engineering," *Artificial Intelligence Review*, vol. 41, no. 3, pp. 317–329, 2014.

[94] A. Martin, S. Emmenegger, and G. Wilke, "Integrating an enterprise architecture ontology in a case-based reasoning approach for project knowledge," in *Enterprise Systems Conference (ES), 2013*. IEEE, 2013, pp. 1–12.

[95] I. Panella, "Artificial intelligence methodologies applicable to support the decision-making capability on board unmanned aerial vehicles," in *Bio-inspired Learning and Intelligent Systems for Security, 2008. BLISS'08. ECSIS Symposium on*. IEEE, 2008, pp. 111–118.

[96] C. Hellert and P. Stütz, "A concept for adaptation of perceptual capabilities for uav platforms using case-based reasoning," in *52nd Aerospace Sciences Meeting*, 2014, p. 0792.

[97] H. Borck, J. Karneeb, R. Alford, and D. W. Aha, "Case-based behavior recognition to facilitate planning in unmanned air vehicles," KNEXUS RESEARCH CORP SPRINGFIELD VA, Tech. Rep., 2014.

[98] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI communications*, vol. 7, no. 1, pp. 39–59, 1994.

[99] H.-D. Burkhard, "Case completion and similarity in case-based reasoning," *Computer Science and Information Systems*, vol. 1, no. 2, pp. 27–55, 2004.

[100] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[101] D. A. Abduljabbar and N. Omar, "Exam questions classification based on bloom's taxonomy cognitive level using classifiers combination," *Journal of Theoretical and Applied Information Technology*, vol. 78, no. 3, p. 447, 2015.

[102] J. Uhrmann and A. Schulte, "Concept, design and evaluation of cognitive task-based uav guidance," *J. Adv. Intell. Syst*, vol. 5, no. 1, 2012.

[103] M. Barak and M. Assal, "Robotics and stem learning: students' achievements in assignments according to the p3 task taxonomy—practice, problem solving, and projects," *International Journal of Technology and Design Education*, vol. 28, no. 1, pp. 121–144, 2018.

[104] M. F. Pinto, A. G. Melo, A. L. Marcato, and C. Urdiales, "Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle," in *Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on*. IEEE, 2017, pp. 1324–1329.

[105] M. Cummings, "Operator interaction with centralized versus decentralized uav architectures," in *Handbook of Unmanned Aerial Vehicles*. Springer, 2015, pp. 977–992.

[106] X. F. F. C. Chart, "Digi," https://www.digi.com/pdf/chart_xbee_rf_features.pdf, 2018, accessed in 10/07/2018.

[107] J. Chen, J. Xie, Y. Gu, S. Li, S. Fu, Y. Wan, and K. Lu, "Long-range and broadband aerial communication using directional antennas (acda): design and implementation," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 793–10 805, 2017.

[108] M. Malik, F. Farahmand, P. Otto, N. Akhlaghi, T. Mohsenin, S. Sikdar, and H. Homayoun, "Architecture exploration for energy-efficient embedded vision applications: From general purpose processor to domain specific accelerator," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*. IEEE, 2016, pp. 559–564.

[109] S. S.-D. Xu and T.-C. Chang, "A feasible architecture for arm-based microserver systems considering energy efficiency," *IEEE Access*, vol. 5, pp. 4611–4620, 2017.

[110] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2016.

[111] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "Uavfog: A uav-based fog computing for internet of things," in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017, pp. 1–8.

[112] H. Lewis and M. Brown, "A generalized confusion matrix for assessing area estimates from remotely sensed data," *International Journal of Remote Sensing*, vol. 22, no. 16, pp. 3223–3235, 2001.

[113] M. B. Bejiga, A. Zeggada, A. Nouffidj, and F. Melgani, "A convolutional neural network approach for assisting avalanche search and rescue operations with uav imagery," *Remote Sensing*, vol. 9, no. 2, p. 100, 2017.

[114] G. Burghouts, A. van Eekeren, and J. Dijk, "Focus-of-attention for human activity recognition from uavs," in *Electro-Optical and Infrared Systems: Technology and Applications XI*, vol. 9249. International Society for Optics and Photonics, 2014, p. 92490T.

[115] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4104–4113.

[116] M. Works, "Matlab user manual version r2016b," *Math Works: Natick, MA, USA*, 2016.

[117] P. Houzé, E. Mory, G. Texier, and G. Simon, "Applicative-layer multipath for low-latency adaptive live streaming," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–7.

[118] K. Bilal and A. Erbad, "Impact of multiple video representations in live streaming: a cost, bandwidth, and qoe analysis," in *Cloud Engineering (IC2E), 2017 IEEE International Conference on*. IEEE, 2017, pp. 88–94.

[119] Y. Zhou, N. Cheng, N. Lu, and X. S. Shen, "Multi-uav-aided networks: Aerial-ground cooperative vehicular networking architecture," *IEEE Vehicular Technology Magazine*, vol. 10, no. 4, pp. 36–44, 2015.

[120] Sunnysky, "X2212," http://www.rcsunnysky.com/content/23.html, 2018.

[121] F. Morbidi, R. Cano, and D. Lara, "Minimum-energy path generation for a quadrotor uav," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1492–1498.

[122] D. H. Choi, S. H. Kim, and D. K. Sung, "Energy-efficient maneuvering and communication of a single uav-based relay," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 3, pp. 2320–2327, 2014.

[123] A. Trotta, M. Di Felice, F. Montori, K. R. Chowdhury, and L. Bononi, "Joint coverage, connectivity, and charging strategies for distributed uav networks," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 883–900, 2018.