TÉCNICO LISBOA

Evaluation of Non-Cooperative Sensors for Sense and Avoid in UAV Systems

Pablo Arroyo Izquierdo

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors : Prof. Afzal Suleman
Prof. Humayun Kabir

Examination Committee

Chairperson: Prof. Fernando José Parracho Lau
Supervisor: Prof. Afzal Suleman
Member of the Committee: Prof. Francisco Fernandes Castro Rego

September 2019

A million dreams for the world we're gonna make

Acknowledgements

In first place, I would like to thank professor Suleman for allowing me to be in Victoria conducting my thesis, as well as to the rest of the team I've shared these six months with. A special word to my research partners António and Luís and to Kabir, who was always asking the right questions. But this thesis is only possible thanks to those people that have help me out during all these years.

Thanks to all those teachers during the school and high school that encouraged me to become the person who I am today.

Thanks to those classmates who became real friends. Special thanks to V, S, L and M.

Thanks to all those people I've met with whom I've shared special moments. You know who you are.

Thanks to those unforgettable experiences I've lived but also thanks to the mistakes I've made.

Thanks to the cinema, thanks to the music and, of course, thanks to the musical movies.

Y gracias a mi familia, especialmente a mis padres, quienes estarán ahí sin importar lo lejos que esté.

Abstract

The number of civil and military applications for Unmanned Aircraft Vehicles (UAV) is increasing in the last years, such as firefighters, search-and-rescue missions or package delivery, among others. Due to their ability of performing a large variety of important tasks with higher manoeuvrability, longer endurance and less risk to human lives, small UAV are particularly suitable.

But to carry out these tasks, it is mandatory to guarantee a safe performance and a correct integration into non-segregated airspace. Integrating unmanned aircraft into civil airspace requires the development and certification of systems for sensing and avoiding (SAA) other aircraft. In particular, non-cooperative Collision Detection and Resolution (CD&R) for UAV is considered as one of the major challenges to be addressed.

The new project Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence started by Boeing at the Center for Aerospace Research (CfAR) requires from definition of the SAA system and a rigorous analysis of it before the system can be developed and eventually certified for operational use.

This paper will be focused on evaluating the capabilities of the non-cooperative sensors for a SAA system, reviewing the different sensors available and the data fusion techniques to merge the information provided by the different sources. Finally, algorithms for visual cameras image processing using machine and deep learning techniques will be developed and compared, with the aim to provide an effective obstacle detection capability.

Keywords: Sense and Avoid ; Non-cooperative sensors; Data Fusion; Machine Learning; Deep Learning

Resumo

Nos últimos anos tem-se verificado um aumento significativo no uso de *Unmanned Aircraft Vehicles (UAV)* para fins civis e militares. Combate de incêndios, procura e salvamento ou entrega de encomendas são alguns exemplos. Devido à capacidade de realizar uma grande variedade de tarefas de forma ágil, com uma autonomia considerável e com baixo risco para o ser humano, pequenos UAV, em particular, são uma escolha confiável.

Contudo, para realizar tais tarefas, é necessário garantir uma segura e correta integração num espaço aéreo que não se encontra segregado. Introduzir *unmanned aircraft* em espaço aéreo civil requere o desenvolvimento e a certificação de sistemas para *sensing and avoiding (SAA)* de outras aeronaves. Em particular, um dos maiores desafios a ser abordado é non-cooperative *Collision Detection and Resolution (CD&R)* para UAV.

O novo projeto *Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence* iniciado pela Boeing em parceria com o *Center for Aerospace Research (CfAR)* exige uma definição do sistema SAA e uma rigorosa análise do mesmo antes de poder ser desenvolvido e eventualmente certificado para ser utilizado.

A presente tese focar-se-á em a avaliar as capacidades de sensores não cooperativos para o sistema de SAA, descrevendo os diferentes sensores previamente disponíveis e as técnicas de fusão de dados usadas para combinar a informação proveniente das diferentes fontes. Por fim, serão apresentados e comparados diferentes algoritmos para processamento de imagem provenientes de visual cameras, que foram desenvolvidos recorrendo a técnicas de machine e deep learning, sendo o principal objetivo de ser desenvolvida a capacidade de detetar obstáculos de forma eficaz.

Palavras-chave: *Sense and Avoid* ; Sensores não cooperativos; Fusão de dados; *Deep Learning* ; *Machine Learning*

Contents

List of Tables **xi**

List of Figures **xiii**

Glossary **xvii**

1 Introduction **1**

1.1 Context and Motivation 1

1.2 Unmanned Aircraft Systems and Sense and Avoid 2

1.3 Objectives 4

1.4 Thesis Outline 4

2 Sense and Avoid System Definition **7**

2.1 Sense And Avoid Taxonomy 7

2.2 Architectures 9

2.3 Sensing 10

2.3.1 Requirements 11

2.3.2 Cooperative sensors 12

2.3.3 Non-cooperative sensors 13

2.4 Detecting 14

2.5 Avoiding 17

2.6 System Definition 18

3 Non-cooperative Sensors **21**

3.1 Why non-cooperative sensors 21

3.2 Radars 22

3.3 LIDARs . . . 24
3.4 Acoustic sensors . . . 26
3.5 EO cameras . . . 28
3.6 Sensor Comparison . . . 31

4 Data Fusion **33**
4.1 Why data fusion . . . 33
4.2 Classification . . . 34
4.3 Data Association . . . 35
4.3.1 K-Means . . . 36
4.3.2 Probabilistic Data Association . . . 37
4.3.3 Joint Probabilistic Data Association . . . 37
4.3.4 Distributed Joint Probabilistic Data Association . . . 37
4.3.5 Multiple Hypothesis Test . . . 38
4.3.6 Distributed Multiple Hypothesis Test . . . 38
4.3.7 Graphical Models . . . 39
4.4 State Estimation . . . 39
4.4.1 Maximum Likelihood and Maximum Posterior . . . 40
4.4.2 Kalman Filter . . . 41
4.4.3 Distributed Kalman Filter . . . 41
4.4.4 Particle Filter . . . 42
4.4.5 Distributed Particle Filter . . . 42
4.4.6 Covariance Consistency Methods . . . 43
4.5 Fusion Decision . . . 44
4.5.1 Bayesian Methods . . . 44
4.5.2 Dempster-Shafer Inference . . . 45
4.5.3 Abductive Reasoning and Fuzzy Logics . . . 45
4.5.4 Semantic Methods . . . 47
4.6 Other solutions . . . 47

5 Machine and Deep Learning for obstacle detection using visual electro-optical sensors **49**
5.1 Artificial Intelligence, Machine Learning and Deep Learning . . . 49

5.2 Convolutional Neural Networks . . . 51
5.2.1 Regions with CNN features . . . 52
5.2.2 Fast R-CNN . . . 53
5.2.3 Faster R-CNN . . . 54
5.3 Support Vector Machine . . . 54
5.4 Implementation of Faster R-CNN for aircraft detection . . . 55
5.5 Implementation of SVM and edge detection for obstacle detection above the horizon . . . 60
5.5.1 Sky segmentation . . . 60
5.5.2 Obstacle detection . . . 62
5.6 Results . . . 62

6 Simulation environment **67**
6.1 ROS . . . 67
6.1.1 ROS Graph . . . 68
6.1.2 ROS Packages, ROS Topics and `rosrun` . . . 69
6.2 Gazebo . . . 69
6.3 Simulation . . . 71
6.3.1 World . . . 72
6.3.2 UAV . . . 72
6.3.3 Data collection . . . 73

7 Conclusion **75**
7.1 Achievements . . . 75
7.2 Future Work . . . 76

Bibliography **77**

A Sensor models **83**
A.1 Radar models . . . 83
A.2 LIDAR models . . . 85
A.3 Camera models . . . 85

List of Tables

2.1 Range and FOR requirements 11

2.2 Single-sensor vs. Multi-sensor SAA architectures 15

3.1 Sensor Comparison. ++: Ideally suited. +: Good performance. 0: Possible but drawbacks to expect. -: Only possible with large additional effort. –: Impossible. n.a.: Not applicable 31

5.1 Machine learning vs. deep learning 51

5.2 Comparison of both implementations 66

6.1 Camera characteristics 72

List of Figures

1.1 Airspace Classes [4] . . . 3

2.1 Basic Taxonomy of SAA systems . . . 8
2.2 Different scenarios depending on the location of the SAA components . . . 9
2.3 Cylindrical collision volume and collision avoidance/safe separation thresholds [7] . . . 12
2.4 Architecture of a tracking system . . . 14
2.5 Sensor fusion configurations . . . 16
2.6 System Definition . . . 18

3.1 Microphone polar plots [10] . . . 27
3.2 Process for object detection in EO cameras . . . 29
3.3 Example of basic edge detection algorithm . . . 30

4.1 Data fusion schema . . . 34
4.2 Example fuzzy membership function [47] . . . 46
4.3 Schema of a camara-based enhancement of the radar measurement . . . 47

5.1 Artificial intelligence, machine learning and deep learning [49] . . . 50
5.2 Convolutional Neural Networks [51] . . . 52
5.3 Regions with CNN features [52] . . . 52
5.4 Fast R-CNN [53] . . . 53
5.5 Comparison of R-CNN and fast R-CNN . . . 53
5.6 Faster R-CNN and RPN [54] . . . 54
5.7 SVM classification (adapted from [55]) . . . 55
5.8 Relevant elements and selected elements . . . 57
5.8 Accuracies and prediction times for different networks (I) . . . 58

5.9 Accuracies and prediction times for different networks (II) 59
5.10 Sky and ground features 61
5.11 Results and detection time for different images 63
5.12 Results and detection time for close obstacles 64
5.13 Results and detection time for far away obstacles 65
5.14 Other obstacle detection with the SVM + edge implementation 66

6.1 Example of a ROS graph 68
6.2 Prototyping 70
6.3 Quadrotor carrying an optical camera and a LIDAR and their information 71
6.4 Simulation environment 72
6.5 Simulated UAV 73
6.6 Algorithms run over images collected from the simulation environment 73

A.1 True View Radar 83
A.2 Demorad Radar 84
A.3 Chemring Radar 84
A.4 360º Sense and Avoid Radar 84
A.5 M8 LIDAR 85
A.6 Ultra Puck LIDAR 85
A.7 Blackfly S USB3 camera 86
A.8 Oryx 10GigE camera 86

Glossary

ADS-B Automatic Dependent Surveillance Broadcast.

AI Artificial Intelligence.

ATC Air Traffic Control.

CCD Charge Couple Device.

CfAR Centre for Aerospace Research.

CMOS Complementary Metal-oxide-semiconductor.

CNN Convolutional Neural Network.

DoD US Department of Defense.

EASA European Aviation Safety Agency.

EKF Extended Kalman Filter.

EO Electro-optical.

FAA Federal Aviation Administration.

FOR Field of Regard.

FOV Field of View.

GCS Ground Control Station.

GPS Global Positioning System.

ICAO International Civil Aviation Organization.

IFOV Instantaneous Field of View.

IFR Instrumental Flight Rules.

IR Infrared.

JPDA Joint Probabilistic Data Association.

KF Kalman Filter.

LIDAR Light Detection and Ranging.

MAP Maximum Posterior.

MHT Multiple Hypothesis Test.

ML Maximum Likelihood.

PDA Probabilistic Data Association.

PF Particle Filter.

PRF Pulse Repetition Frequency.

R-CNN Regions with CNN features.

ROS Robot Operating System.

RPN Region Proposal Network.

SAA Sense and Avoid.

SNR Signal-to-noise Ratio.

SVM Support Vector Machine.

TCAS Traffic Alert and Collision Avoidance System.

UA Unmanned Aircraft.

UAS Unmanned Aircraft System.

UAV Unmanned Aerial Vehicle.

UKF Unscented Kalman Filter.

VFR Visual Flight Rules.

Chapter 1

Introduction

1.1 Context and Motivation

Unmanned Aircraft Systems (UASs) is currently a field that utilizes the leading technologies advances and challenges some of the current regulatory and operational practices of major aviation authorities around the world. For that reason, the interest in the UASs has grown over the last years and lots of applications have appeared in the scene, not only military but also civil or commercial.

Most military UASs are used for surveillance, intelligence, reconnaissance and strikes. The trend in this field is to replace dangerous manned missions for unmanned ones, covering a significant part of warfare activity.

About the civil part, the development of these systems has lead to a world of new opportunities to cover a huge market. The Federal Aviation Administration (FAA) estimated that there will be 30000 *drones* in national skies operating by 2020 [1]. Some of the commercial applications for UAS are categorized into five groups and described at Angelov [2] as follows:

- **Earth science applications**, including remote environmental research, atmospheric monitoring and pollution assessment, weather forecast or geological surveys.
- **Emergency applications**, including firefighting, search and rescue, tsunami/flood watch, nuclear radiation monitoring and catastrophe situation awareness or humanitarian aid delivery.
- **Communications applications**, like telecommunication relay services, cell phone transmissions or broadband communications.
- **Monitoring applications**, including homeland security, crop and harvest monitoring, fire detection, infrastructure monitoring (like pipelines, oil/gas lines, etc.) or terrain mapping.
- **Commercial applications**, in the strict sense of the word, including aerial photography, precision agriculture chemical spraying or transportation of goods and post.

Such a big number of applications has to lead to a big number of companies and research teams to focus on that matter, to focus on what makes UAS different than manned air vehicles and, to be able to do those tasks, what will be required to allow them to operate and be integrated with other users into the civil airspace. One of the most important requirements for enabling these operations is to provide the UAS with a Sense and Avoid (SAA) capability that replaces the manned aircraft capability to see and avoid. Because such systems, that have to deal not only with other aircraft but also with the environment around them (terrain obstacles, buildings, trees, wires, birds, etc), are typically very complex and a high level of safety must be maintained, rigorous analysis is required before they can be certified for operational use and a great effort over the past decade has gone into the definition and aviation requirements for SAA.

The Centre for Aerospace Research (CfAR) in collaboration with Boeing and the National Research Council in Canada is starting a research in the SAA where the system definition, data fusion and artificial intelligence are keywords and problems that need to be addressed.

1.2 Unmanned Aircraft Systems and Sense and Avoid

To introduce the reader to the subject, it is interesting to explore what has the manned aviation in common and what are the differences with an UAS. Some of the ideas of this section have been proposed in Angelov [2] and Fasano et al. [3].

There are different ways to refer to a pilotless aircraft. Some of them are Unmanned Aerial Vehicle (UAV) or popularly, drone. But the most reputable organizations like the International Civil Aviation Organization (ICAO), EUROCONTROL, the European Aviation Safety Agency (EASA) or the FAA, as well as the US Department of Defense (DoD) adopted UAS as the correct and most complete term. The cause is that an UAS is more than just an aircraft. A common configuration not only include the flying air vehicle with the systems onboard (i.e., the unmanned aircraft (UA)), but also the ground control station (GCS), that a pilot could use to operate the aircraft, and the command link between both of them.

The fact that the human pilot is not in the aircraft introduces the first problem: they can't see the surrounding environment to understand what is happening around the aircraft. That is one of the reasons to introduce a SAA system. If the UAS is composed of a GCS, then a link between the pilot in the GCS and the aircraft itself is needed. To manage the command and control issues, the inclusion of onboard system for autonomously controlling the operation has to ensure a predictable performance and safety when the link fails. Therefore, it has to be a part of the SAA system. One might think that to address the lack of seeing and avoiding, a first-person view with onboard cameras transmitting the ground station would be enough, but data link limitations and a constrained field of view (FOV) join the limited ability to discern a target in the video picture and make it a very poor solution. Thus, the SAA methods will need to sense and avoid other aircraft and obstacles, either onboard the UA or with the ground station in the loop.

To classify properly a SAA system, the differential fact that the UAS may also have different missions must be taken into account. Just as manned aircraft, UAS can range from larger aircraft (Global Hawk, Reaper) through midsize aircraft (ScanEagle) to small aircraft (like the ones affordable for common people).

It will be considered as large UA the ones that fly among manned aircraft in controlled airspace. Defined by ICAO, countries must provide Air Traffic Control (ATC) services to aircraft flying in this airspace and a certain self-separation must be ensured by the manned aircraft. These airspaces include en-route high altitude airspace (Class A), terminal airspaces (Classes B, C and D) and lower altitude transition airspace (Class E). It is represented in Figure 1.1.

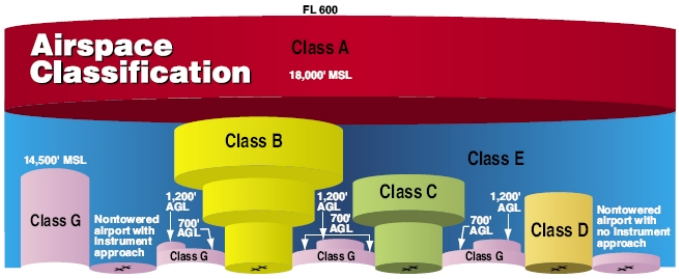


Figure 1.1: Airspace Classes [4]

Small UA have also reached commercial activities and research endeavors, not only recreation and entertainment, and in most of the cases, a certified aircraft and pilot is required to operate safely. The tasks for these purposes are usually developed in uncontrolled airspace, including low altitude Class G airspace.

It can be noted that there will be several differences in a SAA system and architecture depending on the UA size since the environment around them (manned aircraft and other obstacles) and the behaviour of the possible intruders (follow flight rules or not) will be different. On the one hand, a large UA operating in controlled airspace has to deal with manned aircraft as the main obstacle, needs to interact at the levels of separation for those airspaces and has to operate under the appropriate flight rules (instrumental flight rules (IFR) or visual flight rules (VFR)), therefore the avoidance maneuvers have to be the ones expected (right-of-way rule for example). On the other hand, the operating environment for small UA is much different. The mission of these type of UAS most of the times requires operating more closely to obstacles than current aviation separation standards permit. To introduce them to operate safely and predictably in these rapidly evolving environments is then needed new separation definitions and UAS airworthiness requirements.

1.3 Objectives

The final goal of the research developed by the different members of the team in the CfAR is to design a full effective SAA system for autonomous UAVs, able to conduct all the operations onboard while maintaining the demands of that kind of operations. Two constraints appear here and will be discussed throughout this work: the algorithms have to consume few resources and computational time while ensuring good results.

The project Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence was being started as this thesis was being written. Thus, as in all new projects, one of the goals of this thesis is the definition of the problem. A proper architecture has to be defined, the sensors to use have to be selected and the problematic of the multi-sensor data fusion has to be described, attending to the characteristics of the system. Therefore, the first part of the thesis will move in those directions, being the goal to have a better understanding of all these areas to develop the future work in the directions that will be described.

The second part of the work will be focused on the development of an effective detection function for the imagery provided by visual cameras. Machine and deep learning are keywords within the frame of the project at CfAR, thus another goal of the thesis will be to test both techniques and address the advantages and limitations of each one.

Eventually, all this work as well of the work of other members of the team will be merged and for that reason it is needed to find a platform well suited to test different kinds of algorithms, not only to process information of different sensors but also to test path planning algorithms that will use this information to compute a path able to avoid the obstacles sensed. Then, the last goal of the thesis will be to address this problem and to choose the proper platform to do all the operations.

1.4 Thesis Outline

The structure of the thesis is organized as follows:

- **Chapter 2** introduces the basics of a SAA system and ends with a proposed architecture for the work developed at CfAR.
- **Chapter 3** describes non-cooperative sensors, addressing the advantages and limitations of each ones depending on the mission in order to make a choice in which to use.
- **Chapter 4** describes the different steps and data fusion techniques used to fuse the information of different sensors and the characteristics of each one since they have to meet the requirements of the system.
- **Chapter 5** reports the implementation of machine learning and deep learning techniques candidates to be used to develop an obstacle detector for visual camera detection, comparing the

results.

- **Chapter 6** introduces the reader to the simulation environment where all the work developed within the frame of the project conducted at CfAR will be tested.
- **Chapter 7** presents a conclusion for the work and provides suggestions for future work.

Chapter 2

Sense and Avoid System Definition

In this chapter, the basic concepts as well as the functionalities of SAA systems are discussed.

In Section 2.1, the steps and taxonomy of SAA systems are presented. In Section 2.2, different SAA architectures are given. Sections 2.3, 2.4 and 2.5 present a brief and conceptual description about sensing, detecting and avoiding in SAA respectively. Finally, an overview of the proposed system definition and architecture is presented in Section 2.6.

Main sources of information (concepts and ideas) about SAA used to elaborate this chapter are Angelov [2] and Fasano et al. [3].

2.1 Sense And Avoid Taxonomy

The core of SAA systems consists of mainly three steps:

1. **Sense**. Methods for surveilling the environment around the aircraft.
2. **Detect**. To analyze the information provided by the sensing methods to determine whether there is any obstacle around the aircraft and it is a threat.
3. **Avoid**. To choose a suitable action for the aircraft to avoid the threat.

Sensors can be cooperative or non-cooperative. Detection can range from alerting the pilot to estimating the motion and trajectory of the obstacle. The avoidance strategy can range from a basic action (e.g., descend) to some complex actions that consider environmental factors. Both, basic and complex actions, can be remotely operated or achieved by autonomous methods.

Different approaches (architectures and technical solutions) could be taken to accomplish these tasks, and Figure 2.1 shows different approaches.

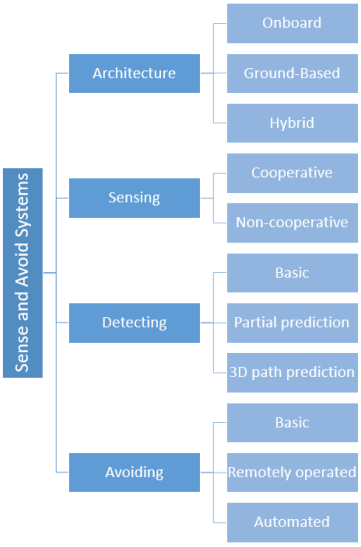


Figure 2.1: Basic Taxonomy of SAA systems

According to the agreements reached at the *Sense and Avoid Workshops* where US FAA and Defense Agency experts discussed a number of fundamental issues [5], an effective SAA system needs to provide two common services. They are a self-separation service that would act before a collision avoidance maneuver is needed, i.e., ensuring that the aircraft remain with a safe separation from each other, and a collision avoidance service to protect a small collision zone and usually achieved by an aggressive maneuver.

To achieve these services, the following list of sub-functions is required as described in Angelov [2]:

1. **Detect** any type of hazards, such as traffic, terrain or weather. At this step, it is merely an indication that something is there.

2. **Track** the motion of the detected object. This requires gaining sufficient confidence that the detection is valid and making a determination of its position and trajectory.

3. **Evaluate** each tracked object, to decide if its track may be predicted with sufficient confidence and to test the track against some criteria that would determine whether a SAA maneuver is needed or not. The confidence test would consider the uncertainty of the position and trajectory. The uncertainty could be great when a track has started, and again whenever a new maneuver has first detected. A series of measurements may be required to narrow the uncertainty about the new or changed trajectory. Also, when a turn is perceived, there is uncertainty about how great a heading change will result in.

4. **Prioritize** the tracked objects based on their track parameters and the tests performed during the evaluation step. In some implementations, this may help to deal with limited SAA system capacity, while in others prioritization might be combined with the evaluation or declaration steps. Prioritization can consider some criteria for the declaration decision that may vary with type of hazard or the context of the encounter (e.g., within a controlled traffic pattern).

5. **Declare** that the paths of the own aircraft and the tracked object and the available avoidance time have reached a decision point that does indeed require maneuvering to begin. Separate declarations would be needed for self-separation and collision avoidance maneuvers.

6. **Determine** the specific maneuver, based on the particular geometry of the encounter, the maneuver capabilities and preferences for the own aircraft, and all relevant constraints (e.g., airspace rules or the other aircraft's maneuver).

7. **Command** the own aircraft to perform the chosen maneuver. Depending upon the implementation of the SAA, this might require communicating the commanded maneuver to the aircraft, or if the maneuver determination was performed onboard, merely internal communication among the aircraft's sub-systems.

8. **Execute** the commanded maneuver.

2.2 Architectures

A preliminary classification of the architectures can be done depending on the location of the information sources (e.g. sensors) and processing and decision making centers [2] [3].

(a) Onboard SAA (b) Ground-Based SAA (c) Hybrid SAA

Figure 2.2: Different scenarios depending on the location of the SAA components

As Figure 2.2 shows, these elements can be located onboard of the aircraft or on the ground. Besides, a hybrid architecture can be used.

- **Onboard SAA** (2.2(a)): Sensing, processing information and its evaluation take place on the UA, by an automated onboard processing capability. This type of system is the most challenging due to size, weight and power of the needed devices to do all onboard. For instance, the sensors have to be light enough to not to compromise the UA payload but at the same time, they have to be able to meet the requirements for an effective SAA system. Then, also the onboard CPU has to be enough powerful to process all the information and to provide a solution to avoid the obstacle. One of the goal in the SAA field is to introduce Artificial Intelligence (AI) algorithms to compute 4D path

planning to make trajectories able to meet the avoidance requirements. Working on optimization of this kind of algorithms is being done for the processors onboard the UA to be able to compute it. All in all, this is the goal to be accomplished by future systems.

- **Ground-Based SAA** (2.2(b)): the three tasks are carried out in the ground. For example, a radar senses the environment and sends this information to the GCS. After a decision is taken, it is sent back to the UA to perform the proper maneuver. This type of configuration may be the simplest one since it is compatible with all kinds of UASs and can work with a few or even no modifications for different UAS designs. However, there is a serious limitation in terms of the area in which the UA can fly, since the sensor is in the ground and it can only sense an area around it.
- **Hybrid SAA** (2.2(c)): While sensing takes place on the UA, detection and tracking of the target may take place both on the aircraft or in the GCS. The information is evaluated in the ground and a maneuver is selected and then transmitted to the UA. This can be a trade-off between the two other configurations.

2.3 Sensing

A surveillance system can be implemented in different ways to accomplish their function of detecting possible hazards. Depending on the choices made, the system will have different capabilities in terms of coverage volume, types of measurements, accuracies, update rates and probabilities of false detection.

In this section, a background about sensing is given. First of all, it is advantageous to describe some of these terms and sensor parameters that characterize a sensor.

- **Range**: distance measured from the sensor within which some good probability of detection of targets may be expected.
- **Field of View (FOV)**: solid angle through which a detector is sensitive to electromagnetic radiation.
- **Field of Regard (FOR)**: total area that a sensing system can perceive by pointing the sensor. For a stationary sensor, FOR and FOV coincide.
- **Update rate**: time between sensor measurements. A good sensor will detect the target at every interval and its effective update rate will be high.
- **Accuracy**: uncertainty of the sensor measurements. It is often given in a single dimension, thus the evaluation must combine accuracy values for different dimensions in order to create a volume of uncertainty.
- **Integrity**: probability that a measurement has the accuracy specified.

It is worth to mention that there is a strong relationship between these terms. An integrated approach when designing the sensing system is then necessary.

2.3.1 Requirements

A SAA system needs to fulfill some requirements in order to ensure the safe performance of the UA in the airspace while providing obstacle detection and tracking. In Table 2.1, the proposed range and FOR requirements for a SAA system are listed. These requirements, in the absence of specific technical airworthiness requirements, are derived from the current regulations applicable to the manned aircraft capability to see and avoid [6].

Detection range requirements [NM]				
Altitude	Manned	UAS		
	Nominal pilot	Autonomous	Line-of-sight	Beyond line-of-sight
Low	2,6	1,1	1,8	1,9
Medium	4,2	1,8	2,9	3,1
High	5,7	2,8	4,1	4,3
FOR requirements [^{o}]				
Azimuth	$\pm$ 110			
Elevation	$\pm$ 15			

Table 2.1: Range and FOR requirements

The basic requirements for sensing range are set to a distance that must give enough time to perform an avoidance maneuver which keeps a minimum separation between the UA and the intruder once it is detected. In fact, detection by itself is not enough to evaluate a possible threat. The collision detection and avoidance maneuvers can only be performed when a confirmed track is present. It occurs at the range named declaration range. Distinction between detection and declaration is then important since an early obstacle detection is needed to provide enough time for executing the trajectory avoidance manoeuvres.

A 500 ft separation is considered as the minimum value to prevent a near mid-air collision. This introduces the concept of a cylindrical collision volume, with a horizontal radius of 500 ft and a vertical range of 200 ft, as it can be seen in Figure 2.3.

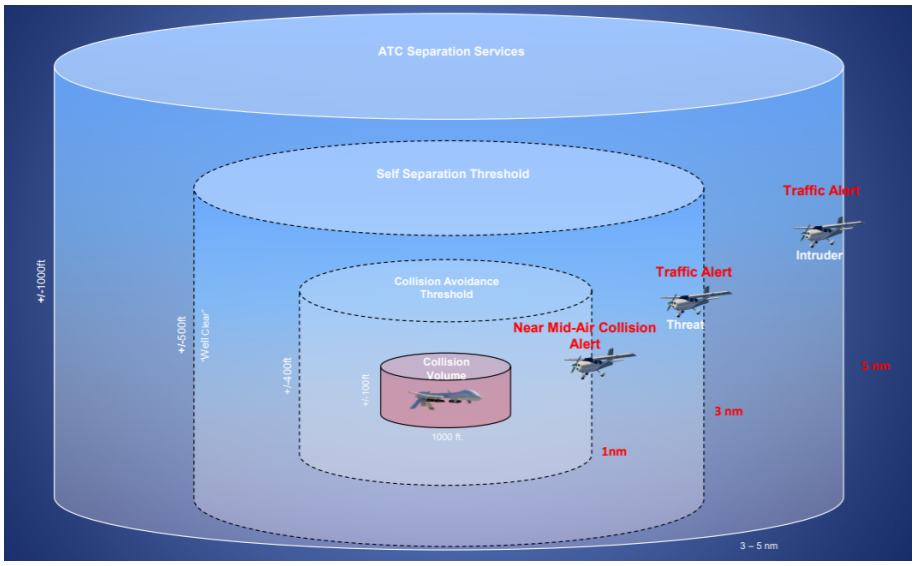


Figure 2.3: Cylindrical collision volume and collision avoidance/safe separation thresholds [7]

A head-on collision is then the worst case to evaluate the detection range.

The requirements for the FOV usually apply for non-cooperative architectures since the cooperative sensors collect information from the whole space around the UA. The FOV has to be equivalent or superior to that of a pilot in the cockpit in order to achieve the equivalent level of safety of a manned aircraft. For a non-gimballed installation, FOV and FOR coincide and the angles needed were shown previously in Table 2.1.

Other essential criteria for designing an effective SAA system is described in Ramasamy et al. [6] as follows:

- Common FOV/FOR for visual and thermal cameras.
- Accurate and precise intruder detection (static and dynamic), recognition and trajectory prediction (dynamic).
- Effective fusion schemes for multi-sensor data augmentation. especially by tight coupling.
- Identification of the primary means of cooperative and non-cooperative SAA system for integrity requirements.

2.3.2 Cooperative sensors

Cooperative technologies are those that receive radio signals from another aircraft's onboard equipment. The cooperative sensors are the most accurate and they provide a satisfactory means of sensing appropriately-equipped aircraft, but they are not able to detect intruder aircraft not equipped with the proper avionics. Cooperative refers to the source of the sense information identifying the locations of other aircraft and specifically refers to transmissions. This kind of sensors, as described in Howell et al.

[8], have better accuracy and integrity of aircraft position data. This improvement in accuracy and integrity leads to simpler conflict detection and collision avoidance algorithms, which in turn reduces the complexity and cost for development and certification. Cooperative sensors also offer significant advantages in reduced size, weight, and power requirements for the UA. A UAS can sense where the intruder aircraft is and take decisions about how to avoid it. The main sensors used in this category are the Traffic Alert and Collision Avoidance System (TCAS) and the Automatic Dependent Surveillance Broadcast (ADS-B).

To respond to ground-based secondary radar interrogations, a large number of aircraft carry a transponder. Indeed, in airspace classes A, B and C aircraft are required to be equipped with a transponder. This technology has been exploited for the manned aircraft TCAS, but it operates independent of the ground-based secondary radar and it provides position information of a potential conflicting aircraft. However, a UAS has different operating characteristics and the TCAS would have to be significantly adapted.

The ADS-B utilizes a navigation source like Global Positioning System (GPS) (thus, it depends on its accuracy and reliability information) and broadcasts the position, velocity and other data without the need of being interrogated. Other aircraft can use the received information and use it to calculate the possibility of collision.

All in all, an UAS could use this type of sensors to keep a safe separation from cooperative aircraft and use non-cooperative sensors for the rest of the aircraft and other obstacles.

2.3.3 Non-cooperative sensors

In airspace where non-cooperative traffic is allowed, other technologies are needed to detect traffic. Non-cooperative sensors detect intruders or other obstacles in the UA FOR when cooperative systems are unable to do that. The process is performed by scanning that region and determine if a measure of the sensor, like the level of energy detected in a specific bandwidth, can be associated to an object that represents a collision threat. It replicates the pilot‘s capabilities of using onboard resources and their senses.

Some examples of non-cooperative sensors are electro-optical (EO), thermal, infrared systems (IR), Light Detection and Ranging (LIDAR), radar and acoustic sensors. While optical and acoustic are best for angular measurement, radar and laser are best for ranging. EO and IR systems are particularly attractive for UA due to their power requirements and payload sizes, smaller than radar systems; and LIDARs are used for detecting, warning and avoiding obstacles in low-level flight. Its angular resolution and accuracy characteristics, as well as its detection performance in a wide range of incidence angles and weather conditions, provide a good solution for obstacles detection and avoidance. Thus, multi-sensor architectures could be better to develop a SAA system although their implementation could be harder. The sensor fusion concept will be discussed in depth later in Chapter 4.

These sensors, in turn, can be classified into two groups: active or passive. Passive sensors use

the energy from other sources, like heat emission or sunlight, to make the measurement, while active sensors provide their source of energy. EO cameras that exploit sunlight or thermal emissions in the thermal IR bandwidth (coming from the aircraft engine for example) and acoustic sensors are passive sensors. Radars and LIDARs that creates their own *pulse* are active sensors. For the passive sensors, the signal travels only a single-way instead of two-way, as in the case of the active sensor,s and no intermediate reflection is needed, which has repercussions on the path loss. However, active sensors use to be more accurate and reliable, although their demanding of resources, such as power, space and weight, is bigger.

More information about non-cooperative sensors will be given in Chapter 3.

2.4 Detecting

Once an object is sensed, a function is needed to detect whether it represents a possible threat or not. It is divided into two sub-functions: tracking and conflict detection. For static obstacles such as ground obstacle, the process is easier. A conflict happens when the distance between the UA and the obstacle breaks the minimal defined separation criterion. In the first step, the system needs to associate successive measurements with a specific target. If the target is dynamic, such as intruder aircraft, the system needs to track it over time. A measurement will be associated with a track if its position agrees with the expected position, within some margin of error that must take into account not only the estimation uncertainties of the sensing systems (noise) but also feasible and unexpected maneuvers that the target could perform. To calculate that position, it uses the previous position plus the estimated velocity times the update interval, usually varying between 1 to 5 seconds. It means that a velocity vector should be developed. In the case of some updates are missed, the tracking function should be capable of maintaining the track for certain time projecting it ahead to an expected position, otherwise, the track would not be correct. Nevertheless, the uncertainty grows in this case.

The general tracking architecture shown in Figure 2.4 can be applied to different SAA systems and operating scenarios [3].

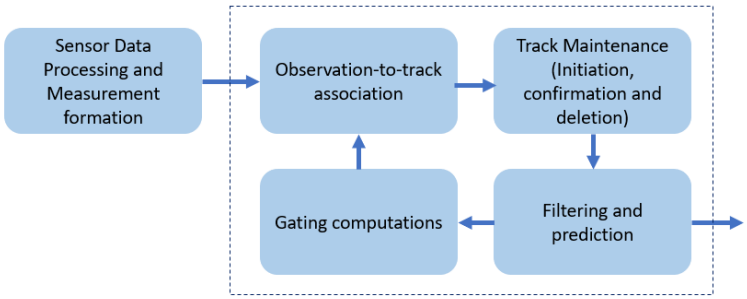


Figure 2.4: Architecture of a tracking system

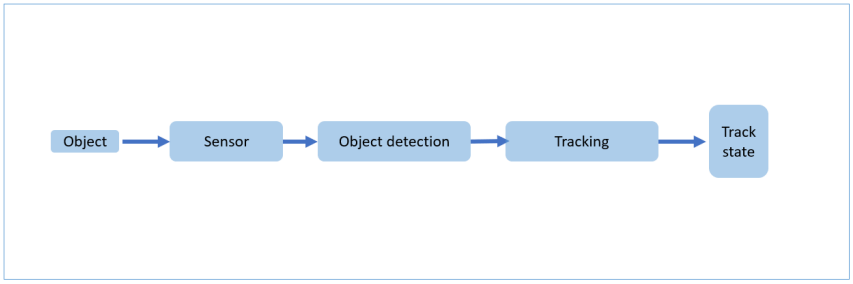
If the tracking system is combining the measurements from different technologies, sensor fusion becomes an important concept. In fact, there are several advantages in using a multi-sensor architecture in terms of the reliability on the systems. They are listed in Table 2.2.

Architecture	Advantages	Issues
Single-sensor	Simpler to implement Reduced impact of misalignment for non-cooperative sensors	Harder design trade-offs to fulfill sensing requirements with only one sensor
Multi-sensor	Improved performance (accuracy, integrity, robustness) Increased sensing range Reduction of computational weight Potential reduction of false detection Relaxed sensing requirements for single sensor in new designs	Complexity of implementation Additional risks of duplicated tracks Impact of residual misalignment for non-cooperative sensors

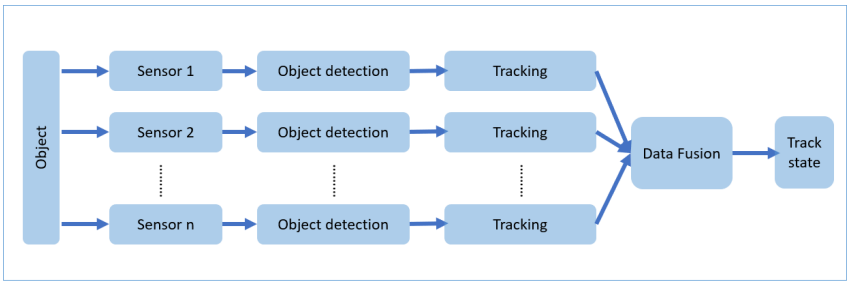
Table 2.2: Single-sensor vs. Multi-sensor SAA architectures

The combination of multiple sensors rises a question on how to combine and process data from all the sensors, taking advantage of their combinations. Data fusion is the answer to this question. It has been studied for many years, especially for robotics applications. Data fusion allows taking advantage of the combination of sensors, increasing the overall sensing performance.

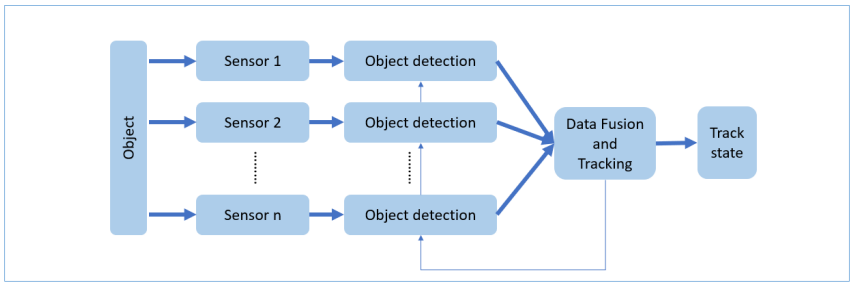
The data fusion might be performed at different stages of the processing. It might be performed on the raw sensor data (Figure 2.5(c)), or after tracking is made(Figure 2.5(b)). When data fusion is performed on the raw sensor data, the detection and tracking stages are performed on the combined data from the sensors. If the data fusion is made after tracking then the process is somehow different, in this case, data from a single sensor is used to identify and track an intruder, after this, the data fusion algorithms combine the tracking files generated from each of the sensors to provide more reliable tracking information.



(a) Single-sensor configuration



(b) Multi-sensor configuration with sensor-level tracking and high-level data fusion



(c) Multi-sensor configuration with centralized tracking and data fusion

Figure 2.5: Sensor fusion configurations

Other literature such as Castanedo [9] gives further parameters to classify data fusion algorithms. The relation between the sensors (input data) used might be complementary (data provided by one sensor is not provided by the other), redundant (same data from different sensors) and cooperative (sensors with different characteristics analyzing the same data). The data fusion process is performed accordingly to some architecture that might be classified as centralized, decentralized or distributed.

In [9], the algorithms are presented in three different categories: data association, state estimation and decision fusion.

- **Data association**. To determine from a set of measurements which ones correspond to each target.
- **State Estimation**. From a set of redundant observations for one target, the goal is to find the set of parameters that provides the best fit to the observed data.
- **Decision Fusion**. These techniques aim to make a high-level inference about the events and activities that are produced from the detected targets.

Another important aspect concerning data fusion is the computational effort needed to accomplish it. There has to be a trade-off between algorithmic performance and computational overhead. Even though there has been a great improvement in the computational power on onboard devices this remains a major concern, specially given what is expected for the Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence project: a fully air-based SAA system, meaning that not-only the data fusion will be processed but all the other stages for SAA operations will also be supported by onboard computers.

Once the sensor fusion is performed and the system has a valid tracked target, a conflict detection function is needed to distinguish threatening from non-threatening traffic or other hazards. Conflict detection logic is based on τ which approximates the time to the closest point of approach as follows,

$$\tau = -\frac{r}{\dot{r}} \tag{2.1}$$

where r and $\dot{r}$ are the relative position and speed between the UA and the obstacle respectively. Note that $\dot{r}$ will be negative when the UA and the threatening obstacle converge. The time is typically compared with a maximum look-ahead time so that intruders with a large time to the closest point of approach can be discarded and considered as non-immediate threats. When the trajectory is not leading to a collision, the system has to check that a certain volume of airspace is not penetrated. Uncertainty and unforeseen maneuvers should have taken into account to add some margin to establish that volume.

2.5 Avoiding

After sensing the environment and detecting a collision risk, the UAS must determine an effective avoidance maneuver, it must plan a new path while avoiding the detected obstacles. This maneuver must take into account aircraft capabilities in accelerating laterally, vertically or changing speed, the ultimate climb or descent rates or bank angle to be achieved; constrains deriving from airspace rules, compatibility with avoidance maneuvers to be performed by an intruder aircraft (expected to follow right-of-way rules), other detected hazard (terrain or other intruder aircraft), errors such as the ones in position measurement (the maneuver would seek the sum of the desired separation plus an error margin) and latencies involved in deciding, communicating and executing the maneuver.

For an autonomous SAA, before selecting a maneuver to be performed by the UA, an algorithm has to evaluate different alternatives on several dimensions of environment and safety. Factors for deciding on an avoidance maneuver must be built into the avoiding and path planning algorithm. Researchers have explored possible decision-making strategies that include when to start the maneuver (while the UA is well clear or squeeze out of the time) and the deviation that should be taken (minimum to optimize flight or more to provide a safety margin). It also needs to use the information available to provide with an avoidance solution that fits the time frame given. Note the differences between an avoidance maneuver if the target is suddenly sensed in an imminent collision or if it is sensed well in advance. Due to that, the key is to design a range of different avoidance maneuvers to ensure safety and to be able to face

different scenarios.

2.6 System Definition

After all the review presented in this chapter, the proposed system architecture for the Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence project developed at CfAR is represented in Figure 2.6. The architecture is designed for a quadrotor UAV.

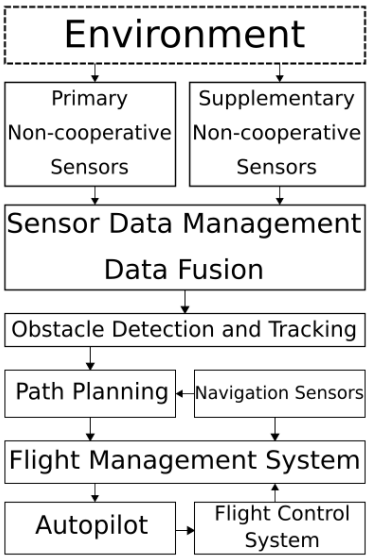


Figure 2.6: System Definition

The process starts with an environment scanning. The scanning will be conducted by a **primary set of non-cooperative sensors** and then supplementary environment information can also be acquired by a **supplementary set of non-cooperative sensors**. This guarantees a fast scanning with the primary set of sensors. The supplementary ones add some redundancy and robustness to the system, especially in situations where the primary set of sensors might not be able to operate. Chapter 3 will discuss the characteristics of different sensors.

The data collected from the sensors is then analyzed and merged in the **Sensor Data Management and Data Fusion** subsystem. In this way, the information provided by each sensor can be combined, increasing the fidelity and robustness of the SAA system. This block is crucial to overcome the limitations of each non-cooperative sensor by using fusion techniques that will be described in Chapter 4.

The data is sent to the **Obstacle Detection and Tracking** subsystem where different tasks are conducted. The tasks include the mapping of each obstacle and prioritizing the ones to avoid. After tracking an intruder, some estimation of its trajectory is done and a list of priorities is set. Based on this list, avoidance actions are taken and sent to the **Path Planning** framework. Here, a trajectory is

generated and optimized. This trajectory is fed with information from the **Navigation Sensors**, which are composed of a GPS to indicate the position of the UAV, an Inertial Measurement Unit (IMU) to determine the attitude and acceleration of the aircraft, an altimeter to calculate the height and a magnetometer to determine the direction of the UAV using the Earth's magnetic field as a reference.

The trajectory and attitude information is transmitted to the **Flight Management System**, which, with complementary information (e.g. battery state, sensor health state), sends a detailed flight plan to the **Autopilot**, which will control the actuators, and thus the UAV kinematics and dynamics, with the aid of the **Flight Control System**.

An initial mapping of the environment is an advantageous technique. A preliminary estimation of static objects can be conducted with a LIDAR and a Simultaneous Localization and Mapping (SLAM) algorithm. This map is then periodically updated when new information is received. The actions referenced above are conducted in a loop to allow quick changes in the trajectory and emergency avoidance maneuvers. Concurrently, information about the UAV status is being transmitted, using telemetry, to a ground station. This station is then capable of overriding the autopilot commands in case, for example, the team wants to abort the mission and land safely.

Chapter 3

Non-cooperative Sensors

In this chapter, the different options for non-cooperative sensors are described.

After justifying why these kind of sensors are chosen in Section 3.1, an overview of them is given. Section 3.2 is focused on radars, Section 3.3 on LIDARs, Section 3.4 on acoustic sensors and Section 3.5 on EO cameras. Finally, Section 3.6 provides an overall view through a comparison of the main sensors described.

The main source of the concepts and ideas for the development of this chapter is Fasano et al. [3].

3.1 Why non-cooperative sensors

As it has been explained in Sections 2.3.2 and 2.3.3, there are two main types of sensors, cooperative and non-cooperative. Any application for SAA needs non-cooperative sensors since the intruder could not be a well-equipped aircraft and, in that case, a cooperative sensor will not detect it.

The application studied in this paper would be initially performed by a small UAV in lower altitudes where non-cooperative traffic is allowed and where ground obstacles could also be important to take into account and, for that reason, to introduce non-cooperative sensors in the architecture takes more importance than ever.

But cooperative sensors could be implemented in these circumstances as an alternative source of information to significantly increase the accuracy of identifying cooperative aircraft. However, from simulation and experimental point of view, the implementation of these kinds of sensors in the SAA architecture would increase the complexity of the problem since testing them would require more than an aircraft flying at the same time and equipped with the same type of cooperative sensor.

The implementation of an effective SAA architecture with only non-cooperative sensors is one of the main topics in today‘s research in the field of UASs and SAA, and for this reason and the ones mentioned above, this paper is also focused on it.

3.2 Radars

One of the solutions for sensing the environment is the radar. A simpler emitter can be a dipole connected to an oscillator with a very good phase and frequency stability, and each time the electrons in this dipole change the sign of their linear momentum, an electromagnetic wave is emitted in the form of a photon.

A typical configuration of radar consists in a monopulse radar, a radar with a single antenna that emits electromagnetic waves modulated by pulses at a certain frequency; or a continuous wave configuration, with two antennas, transmitting and receiving one. A monopulse radar switches the antenna from the transmitter to the receiver after emitting the pulse, employing the diplexer. When the initial pulse hit an object, depending on the reflective properties of its surface, a wave is backscattered to the radar antenna. If it has enough energy, the presence of the object is detected by the radar. The relative range is measured through the travel time (roundtrip) of the wave. On the other side, for a continuous wave radar, Doppler processing carries with the process of the measurement of the range rate. But for a SAA system this type of configuration, that requires two antennas, is much more demanding in terms of size and weight and for that reason, it is usually discarded since the SAA application require compact configurations to be installed onboard the UA. Therefore, to measure the range it is used Equation 3.1,

$$R = c\,\frac{t_{\text{rx}} - t_{\text{tx}}}{2} \tag{3.1}$$

where c is the speed of light, t_{rx} is the time when the pulse is received back in the antenna and t_{tx} is the time when the pulse is transmitted.

Some other parameters to take into account are described in the following lines. One of them is the pulse repetition frequency (PRF). False objects can be detected if this parameter is not selected properly. If a pulse is transmitted before the previous one is received, the value of t_{rx} will correspond to the previous pulse and not to the actual one. For that reason, the PRF needs to be smaller than the reciprocal of the time needed for a pulse to travel a distance that is the double of the maximum range that is assigned to the region that forms the radar detection coverage area.

$$PRF \leq \frac{c}{2\,R_{\text{max}}} \tag{3.2}$$

Common values for PRF range from 3 kHz for R_{max} = 50 km to 30 kHz for R_{max} = 5 km.

The detection of an object is possible if the power of the pulse received by the antenna is considerably greater than the noise associated with the receiver. This is measured through the signal-to-noise ratio (SNR) and the concept described can be expressed with Equation 3.3,

$$\frac{S}{N} \geq \epsilon_{\text{th}} \tag{3.3}$$

where ϵ_{th} is a proper threshold. False alarms (noise misclassified as an object) and missed detections (object misclassified as noise) are a function of this threshold and it is usually mapped by the manufac-

turers. The threshold can be adapted according to the application, for example, by reducing it when the distance from the ground is high enough. Usually is preferred a certain amount of false alarms that the same amount of missed detections, since the first ones can be more easily removed. It can be done even with a single scan exploiting an altitude threshold (removing all detected objects bellow certain altitude known as ground clutter), in which case the radar processing unit needs to know aircraft altitude and attitude; and a velocity threshold (removing stationary objects that are likely to be on the ground using Doppler signature), a more critical process due to the danger of deleting also balloons or hover helicopters. Values of ϵ_{th} are expressed in the decibel scale. The minimum level is 6 dB that corresponds to $\epsilon_{\text{th}} = 3,98$. Higher the value, higher the quality of detection.

The SNR can be computed combining the next expressions. On one hand,

$$S = \frac{P_{\text{tx}}\, G^2\, \lambda_c^2\, \sigma}{(4\pi)^3\, R^4\, L} \tag{3.4}$$

where

- P_{tx} is the power emitted by the transmitter.
- G is the gain of the antenna, for a parabolic one:

$$G = (\frac{\pi\, d}{\lambda_c})^2 \eta_A \tag{3.5}$$

  where d is the diameter of the dish and η_A is the aperture efficiency, dependent on the quality of the antenna manufacturing (typical values between 0,55 and 0,70).
- λ_c is the wavelength associated with the carrier.
- σ is called the radar cross-section and expresses the reflectivity of the object and depends on the type of the object, the material, its relative pose with the own aircraft and the radar λ_c. The typical limit for detection applied for SAA is σ = 1 m^2.
- R is the range.
- L is the transmission loss, mainly caused by atmospheric scattering and absorption. These type of effects have a higher impact on high frequencies.

On the other hand,

$$N = k\, T\, B \tag{3.6}$$

where k is the Boltzmann constant (1,38 10^{-23} J/K), T is the temperature (typically assumed T = 290 K) and B is the radar bandwidth, measured in Hz.

Combining equations 3.4 and 3.6, the SNR is expressed as follows:

$$\frac{S}{N} = \frac{P_{\text{tx}}\, G^2\, \lambda_c^2\, \sigma}{k\, T\, B\, (4\pi)^3\, R^4\, L} \tag{3.7}$$

Equation 3.7 can be used to estimate the maximum distance where an object can be detected. In the extreme case $S/N = \epsilon_{\text{th}}$ and σ is equal to the minimum radar cross-section to be detected. Then, by the substitution of these terms in the SNR equation 3.7, that range can be obtained as follows:

$$R_{\text{det}} = \sqrt[4]{\frac{P_{\text{tx}}\, G^2\, \lambda_c^2\, \sigma_{\text{min}}}{k\, T\, B\, (4\pi)^3\, \epsilon_{\text{th}}\, L}} \tag{3.8}$$

Best performance is provided when a large wavelength and a large pulse width is assigned. A trade-off has to be made for small UA since larger wavelengths need larger antennas. Note that it has a direct influence in the radar carrier frequency f_c. Larger aircraft with 1 m or more diameter antennas can use X band radars (f_c = 10 GHz) while small aircraft need to use Ka-band or W band radars (f_c = 35 GHz or 94 GHz respectively) and they need signal amplification to have enough emitted power.

To cover the required FOR, the radar head can be rotated mechanically or electronically. Mechanical scanning is done with a motorized gimbal structure that allows full rotational control over the radar head, with what this entails, i.e. high complexity of the layout, expensive maintenance, slowness of rotations and difficulties on the installation due to the moving parts and their size. Electronic scanning implies coupling several dipoles on a single strip. With the proper phase difference assigned to each dipole, the relevant figure of interference produces a controlled rotation of the main lobe. Then, beam steering can be realized at high speed without the need for moving parts, but this technique is expensive and requires state-of-the-art technologies. Also, it only can reach an angular span of 50º with a single electronic antenna due to the deformation of the lobes that can increase the risk of receiving a signal from side lobes, so the problem becomes more complicated for larger angular scan widths.

All in all, radars are a good solution for non-cooperative sensing and some of their advantages are:

- They can estimate all terms of the track state.
- The reliability of detection and initial detection distance can be regulated by selecting the level of transmitter power.
- They can deal with bad weather conditions since the degradation is well modeled and it can be well compensated for increasing the emitted power or with the proper carrier frequency.

Nevertheless, they are quite demanding in terms of onboard resources, cost, size, weight, required electric power; thus could be not suitable for small UAV.

3.3 LIDARs

The LIDARs are sensors that follow the same principles that radars but, instead of using microwave energy emitted by the antennas, they use the light emitted by a laser. A laser is a device that uses an effect of quantum mechanics, induced or stimulated emission, to generate a coherent beam of light both spatially and temporally. Spatial coherence corresponds to the ability of a beam to remain a small size

when transmitted through vacuum over long distances, which makes it also a collimated source, and temporal coherence relates to the ability to concentrate the emission in a very narrow spectral range. A typical laser consists of three basic operating elements. A resonant optical cavity, in which light can circulate, usually consisting of a pair of mirrors. One has high reflectance (close to 100%) and the other, known as a coupler, has a lower reflectance and allows the laser radiation out of the cavity. Inside this resonant cavity, there is an active medium with optical gain, which can be solid, liquid or gaseous (usually the gas will be in a partially ionized plasma state). This medium is in charge of amplifying the light. It is where the excitation processes occur and it can be made of many different materials. It is the one that determines to a greater extent the properties of laser light, wavelength, continuous or pulsed emission, power, etc.. Typical gain mediums are He-Ne or Nd-Yag. To amplify light, this active medium needs a certain amount of energy, commonly called pumping. The pumping is generally a beam of light (optical pumping) or an electric current (electric pumping).

In most lasers, the laser starts with the stimulated emission that amplifies the randomly emitted photons spontaneously present in the gain medium. The stimulated emission produces a light that equals the input signal in wavelength, phase and polarization. This, combined with the filtering effect of the optical resonator, gives the laser light its characteristic coherence and can give it uniform polarization and monochromaticity, i.e., all photons emitted having almost the same wavelength or frequency. After being emitted from the source, the light hits a mechanism that supports the scanning of the FOR. This mechanism can be a movable mirror or prism that allows controlling the orientation of the beam boresight axis (axis of maximum gain). An important issue to take into account is that lasers for LIDARs must be eye-safe since the beam could enter through the cockpit of a manned aircraft and hits the pilot‘s eye. 1000 nm lasers are preferred because their frequency is not harmful to the human eye.

Once the system has been described, it is interesting now to introduce the term of the resolution, as well as interesting for the radars and other sensors. The resolution of a sensor is the smallest change it can detect in the quantity that it is measuring, i.e., the minimum relative distance that permits it to discriminate between two objects that are in the beam. Resolution can be defined for all terms included in the track state: range, range rate, angles and angular rates, for example. Thus, all objects identified closer than the resolution are considered as one. Then, the domains of the track state can be divided into equal subdomains called cells, equal to or larger than the resolution. These cells will contain no more than a single object. The angular resolution, that describes the ability of the sensor to distinguish small details of an object, is given by Equation 3.9,

$$\Delta\theta = 1,22\frac{\lambda_c}{d} \tag{3.9}$$

where $\Delta\theta$ is the angular resolution (radians), λ_c is the wavelength of light, and d is a characteristic size. The factor 1,22 is derived from a calculation of the position of the first dark circular ring surrounding the central Airy disc of the diffraction pattern. For typical applications, Nd-Yag lasers with a peak wavelength of 1064 nm (infrared) are adopted. With $d = 5\ 10^{-3}$ m, equation 3.9 becomes in $\Delta\theta = 1,5\ 10^{-2}$ º, which means that the footprint of the beamwidth covers a circular area with a diameter less than 5 m

at a distance of 10 km from the emitter. Then, a huge number of cells required would lead to a very slow scanning process. Some strategies, such as not to transmit pulses for all cells, can be adopted, but it would determine a risk of detecting the object with a considerable delay with respect to the theoretical initial distance that it could be detected. Another problem has been identified: due to the small wavelength, it is very susceptible to atmospheric attenuation effects, that can reduce the maximum detection range to less than 1,5 NM. Due to these constraints, LIDARs as a primary sensor can only provide a feasible solution for SAA in short range operations, such as the ones executed by small fixed wing UAV or multirotor, carried out in airspaces where the velocity is very limited. Other solution would be to use them as secondary sensors in conjunction with EO cameras in order to perform the determination of range and range rate for close objects.

All in all, LIDARs are compact enough to be suitable for small UAV operations but only if they fly in airspace classes such as Class G (see Figure 1.1).

3.4 Acoustic sensors

Acoustic sensors are usually microphones, that measure the level of acoustic pressure emitted by an object, such as the engine of an intruder aircraft. The measurement is done by sensing the pressure difference over the two side faces of a thin diaphragm.

Microphones can be classified according to their polar patterns. On the one hand, an omnidirectional microphone is usually a pressure-operated microphone, i.e., the diaphragm, which picks up sound vibrations in the air, is completely open and exposed to open atmosphere at one side, but completely closed at the other side, that contains a closed volume with a reference constant pressure. The sound vibration is either pushing the diaphragm against the fixed pressure of the air on the other side, or it is reducing the pressure on the front of the diaphragm allowing the pressure behind to push it out. Omnidirectional means that it is sensitive to pressures incoming from any direction. On the other hand, a bidirectional microphone is usually a pressure-gradient microphone, i.e., it has both sides of the diaphragm fully open to the atmosphere, then, it compares the pressure of the sound wave on one side with the pressure of that same sound wave after it has traveled through to the other side. The difference in pressure between the front and the back of the diaphragm depends on the angle of incidence of the sound wave. If the sound arrives from the side of the diaphragm, the pressure will be the same at the front and the rear, therefore the diaphragm will not move and there will be no output. If the sound arrives from a direction normal to the plane of the diaphragm, it will reach its maximum displacement. Bidirectional means that it is sensitive to pressures incoming from a direction determined by the main lobes in the polar plot.

The polar plot determines the gain of the microphone in a certain direction.

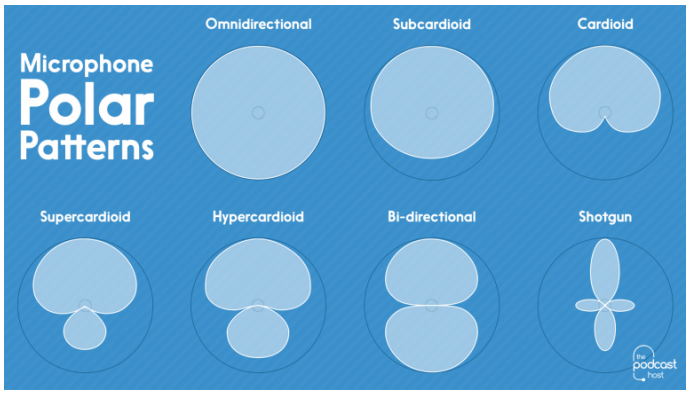


Figure 3.1: Microphone polar plots [10]

Figure 3.1 shows, as said before, that an omnidirectional microphone has the same gain in all directions whereas the bidirectional has two main lobes with a phase shift of 90º, therefore it is sensitive to sounds incoming from azimuths 0º and 180º and it is deaf to sounds from azimuths 90º and 270º. Both types of microphones can be combined to form other polar plots such as cardioid, hypercardioid or shotgun. Cardioid is a form of microphone with a poor directivity but a great angular dispersion of the main lobe. It has the maximum gain at 0º and a null at 180º. Hypercardioid and shotgun are configurations where the proportion lies in favour of the bidirectional microphone, in order to increase the directivity. To determine the bearing of the intruder, a SAA system would need a directional microphone. Also, a proper installation strategy must be considered to reduce the impact of disturbance sources as the own UA or other unwanted sources and to protect the microphones against dirt and other things that could degrade them.

To estimate the range within which the acoustic sensor can detect an intruder, Equation 3.10 is used

$$R_{det} = 10^{\frac{L_w - 11 - DI}{20}} \tag{3.10}$$

where L_w is the sound power level (dB) and DI is the directivity index in the decibels domain given by:

$$DI = 10\, log_{10}(Q) \tag{3.11}$$

The directivity index Q is the equivalent to the gain G in the radar equations, then, for an isotropic (omnidirectional) source $Q = 1$, since it measures the relation with respect to this kind of source, and therefore DI=0. If a jet engine is considered a isotropic source with $L_w = 90$ dB, $R_{det} \approx 9000$ m. A further reduction could be derived by applying parameters such as atmospheric absorption, a linear function of R, and attenuation due to wind and turbulence, very complex to calculate that range from 0 dB for calm weather to 15 dB, according to a rule of thumb.

There are also a lot of noise sources that are not a threat. In addition, they can disturb the transmission of the intruder‘s sound and they are very difficult to predict. Even the speed of sound is dependent

on the thermodynamic properties of the material that transmits the wave, which leads to echoes, distortions and nonlinearities, and, in turn, to false alarms or missed detections. Thus, the recognition of the threat is performed by estimating the spectral signature of the sound received in order to perform a correlation analysis with the spectral signature of an aircraft engine, but problems in the transmitted sound can persist anyway.

For all these reasons, acoustic sensors are strongly dependent on the type of engine and environmental conditions and they are not a good option as a primary source of reference for a SAA system. However, they can be a low-cost option as an auxiliary source.

3.5 EO cameras

EO systems are systems that use a combination of electronics and optics to generate, detect, and/or measure radiation in the optical spectrum, generally in the visible and IR wavelengths. An array of detectors, called pixels, are capable of acquiring the visible intensity of light over a large solid angle, which extent is the FOV, when placed in the focal plane of a suitable lens system. The angular dimension of the FOV in a certain direction is given by the Equation 3.12

$$\Delta\theta = 2\ arctan(\frac{f}{2d}) \tag{3.12}$$

where d is the size in that direction of the detector array and f is the focal length of the lens system. Each pixel also has its own angular width called instantaneous field of view (IFOV), given by Equation 3.13

$$\Delta\alpha = \frac{\Delta\theta}{n} \tag{3.13}$$

where n is the number of detectors distributed along that direction. As there are an array of detectors, each sensor has a vertical and horizontal FOV and IFOV.

For the visible wavelength sensor arrays, there are two types of technologies available: charge couple devices (CCDs) and active pixel sensors with complementary metal-oxide-semiconductor (CMOS). CCDs have a better SNR but are more difficult to integrate with current processing electronics. CMOS technology is the opposite and, in addition, it allows local amplification of poorly lit areas because the light intensity level can be read, but the sensor stills picking up charges. Another thing to take into account related with the SNR is that the Bayer filter reduces the SNR of a sensor, therefore, for normal applications panchromatic cameras are usually preferred to color ones, which output information about the color of a pixel by applying a suitable Bayer filter in front of the array.

In relation with IR sensors, they can be:

- Near IR sensors $\lambda \in [750, 1400]\ nm$, used to enhance vision in poorly lit conditions, benefiting from the quantity of energy emitted by artificial illumination sources. Their detectors are CCDs without coating filters so they collect light and noise over a wide frequency range.

- Thermal IR sensors medium $\lambda \in [3000, 8000]\ nm$ and long $\lambda \in [8000, 15000]\ nm$ wave IR. Essentially, they detect the heat emission from the engines and the exhaust from the plume because, at these wavelengths, there is a peak of the blackbody radiation curves. In particular, medium wave IR allows bright images of hot objects with the adoption of compact cameras. However, forced cooling is needed to keep the level of noise in a reasonable value.

Visible and near IR CCD and CMOS can have up to 5000 pixels per line, thus with a FOV of 50º, using 3.13, it results to an angular pixel size up to 0,01º. For the medium wave IR cameras, that have up to 640 pixels per line, it results in an angular pixel size up to 0,08º, almost one order of magnitude worse. That and the worse noise level make medium wave IR cameras a not suitable option for SAA even though they could support night-time operations.

Figure 3.2 shows the process for image acquisition and detection.

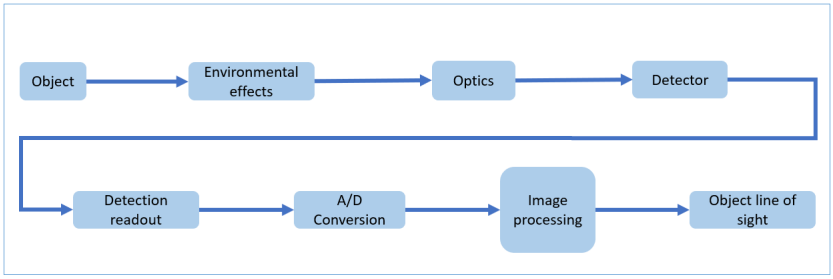


Figure 3.2: Process for object detection in EO cameras

Regardless that the readout (two-dimensional signal in the array into a one-dimensional stream) and the analog to digital (A/D) conversion are performed by electronics embedded in the detector, the image processing is different depending on the application. Even in the same image, it is different for SAA applications due to the regions located above and below the line of horizon have different properties. For instance, the distribution of intensities above the horizon tends to be much more uniform than the ones of pixels below the horizon and for that reason, the SNR tends to be higher below the horizon to detect the same object with similar visibility and illumination conditions. Some classical approaches for the image processing part are:

- **Basic edge detection.** This method is intended to identify points in a digital image where the brightness of the image changes abruptly or, more formally, has discontinuities. Points where the brightness of the image changes sharply are typically organized into a set of curved line segments called edges. Since an object of interest tends to be brighter or darker than its background, the edge detection process will likely highlight its border pixels. Nevertheless, it can generate a high number of false alarms if a thresholding strategy is not adopted. A simple algorithm of this technique, running in Matlab, and its result are shown in Figure 3.3

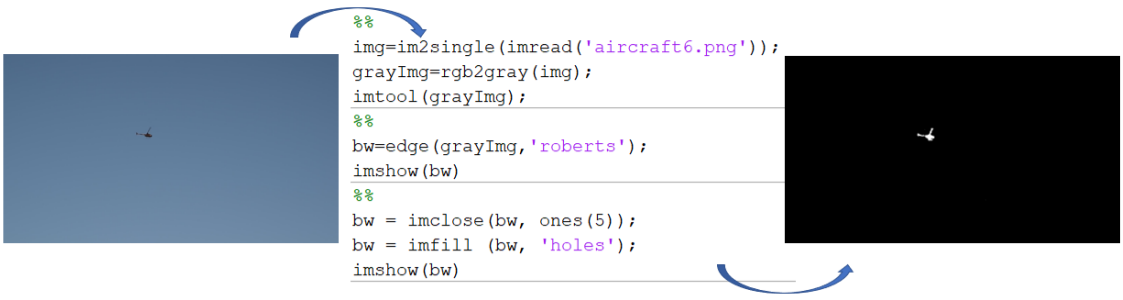


Figure 3.3: Example of basic edge detection algorithm

- **Template matching.** This technique is made through comparing the image acquired by the sensor with a database of expected images of the object of interest. If there is a correlation, the object can be detected. Thus, the objects must have a small variability in terms of shape for the good performance of this method, because it tends to be degraded if the number of objects in the data base is large.

- **Morphological filtering.** It applies specific morphological operators in an appropriate sequence to highlight and separate candidate objects to be detected. The image is binarized by selecting a proper thresholding strategy and the clusters that have the expected size and pixel distributions are classified as objects.

About the performance of the EO cameras, Equation 3.14 for angular span γ is used to estimate the elevation with respect to sensor axis and the azimuth,

$$\gamma = arctan(\frac{f\ n}{d\ q}) \tag{3.14}$$

where q is the number of pixels between the object and the pixel aligned with the boresight axis and d/n is the size of a single detector in the array. For the elevation, the direction of the line in the array used is vertical and for the azimuth horizontal. In general, a detected object is extended over several pixels but its center can be computed with subpixel accuracy by averaging the position of pixels weighted with the measured intensities. Regarding the detection range, two parameters must be taken into account. One is the radiometric detection, referred in Equation 3.3, that needs to be checked for a single pixel. If the measured level is in the order of the level noise, no object can be detected. Nevertheless, the level of sunlight reflected by most aircraft or the heat emitted by the engine of an aircraft is always much greater than the noise of the sensor in practical application and, consequently, radiometric detection is not the one that limits the detection range. The other parameter is the geometric detection, stating that an object can be detected if it is distributed over an area of pixels larger than a minimum. A practical approach requires that the object is extended over 4 or 5 pixels. Then, the detection distance can be computed as follows:

$$R_{det} = \frac{l\ f}{q\ m} \tag{3.15}$$

where l is a characteristic size (such as wingspan), q the minimum number of pixels needed for the detection and m the pixel size. Applying Equation 3.15, an aircraft with a wingspan of 9 m can be

detected by a 1 mega pixel EO camera with focal length of 2 mm and a pixel size of 5 μm at a distance R_{det} = 720 m, with a 5-pixel criterion.

In conclusion, EO cameras are a good primary source of information for small UAV flying into class G airspace with the proper weather conditions and can be combined as a secondary source with other sources such as radars or LIDARs to increase the overall angular resolution and the data rate.

3.6 Sensor Comparison

It can be concluded that the best sensors to be used in a SAA application are radar, LIDAR and EO cameras. For that reason, an overview of their characteristics and limitations is summarized in Table 3.1.

	Short Range Radar	Long Range Radar	LIDAR	Video Camera	3D Camera	Far IR Camera
Range Measurement $<$2 m	0	0	0	-	++	-
Range Measurement 2 - 30 m	+	++	++	-	0	-
Range Measurement 30 - 150 m	n.a.	++	+	-	-	-
Angle Measurement $>$10 deg	+	+	++	++	+	++
Angle Measurement $>$30 deg	-	-	++	++	+	++
Angular Resolution	0	0	++	++	+	++
Velocity Information	++	++	–	–	–	–
Operation in Rain	++	+	0	0	0	0
Operation in Fog or Snow	++	++	-	-	-	0
Operation if Dirt on sensor	++	++	0	–	–	–
Night Vision	n.a.	n.a.	n.a.	-	0	++

Table 3.1: Sensor Comparison. ++: Ideally suited. +: Good performance. 0: Possible but drawbacks to expect. -: Only possible with large additional effort. –: Impossible. n.a.: Not applicable

So, in terms of implementation, the sensors have the following characteristics.

Camera

- Highest resolution
- Long range
- Good angular resolution
- Clear sky / day time solution
- No distance or speed detection
- Low cost solution

LIDAR

- High resolution
- Short range
- Good weather solution
- Angular and distance information
- Expensive equipment

Radar

- Low resolution or single point
- Long range
- All weather solution
- Distance and speed information (rough angular information)
- Expensive equipment

As an example, a configuration with these three sensors could provide information about obstacles by the next means.

- **Camera**. Image (video frames) processed by an algorithm in order to detect obstacles in clear sky providing relative elevation and azimuth.
- **LIDAR**. Point cloud data processed using an algorithm to detect objects (reflective clusters) providing the size and distance of the object as well as angular position. Two sets of data can be compared to calculate the speed based on aircraft information.
- **Radar**. The detection of the object will provide distance and speed (Doppler processing needed) as well as angular position in the case of array radar.

Some cameras, radar and LIDAR models available in the market that could suit the application are shown in Appendix A.

Chapter 4

Data Fusion

The goal of this chapter is to present the different options (architectures and algorithms) for data fusion, their advantages and their limitations.

In Section 4.1, the idea of the data fusion is introduced. Section 4.2 presents the main architectures for a data fusion system. Sections 4.3, 4.4 and 4.5 describe the steps in a data fusion process (data association, state estimation and fusion decision, respectively) and different algorithms used in each one. Finally, in Section 4.6, a way to take advantage from a multi-sensor configuration is provided.

The main sources of the concepts and ideas for the development of this chapter are Castanedo [9] and Fasano et al. [11].

4.1 Why data fusion

Data fusion is the process of combining information from several different sources to provide a robust and complete description of an environment or process of interest.

Data fusion is especially important in any application where large amounts of data must be combined, merged and distilled to obtain information of adequate quality and integrity on which decisions can be made. But also in applications where automated data fusion processes allow the combination of measurements and essential information to provide knowledge of sufficient richness and integrity to formulate and execute decisions autonomously. Data fusion finds application in many military systems, civil surveillance and monitoring, process control and information systems. Data fusion methods are particularly important in the drive towards autonomous systems in all these applications.

Thus, data fusion becomes a key element when dealing with different sensors, different sources of information with different characteristics. For a configuration with a radar, LIDAR and visual EO camera, a schema such as the one in Figure 4.1 is proposed.

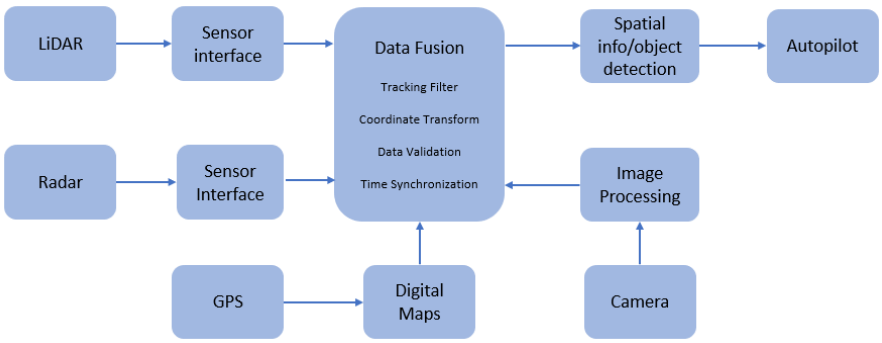


Figure 4.1: Data fusion schema

In order to prepare the raw data to feed into the data fusion processor, to adapt the sensor interface to standard interface (IP, serial, etc.) and to prevent the processing power to be spent on data translation, a block between each sensor and the data fusion itself is needed. Therefore, these blocks can perform primary processing and reduce the complexity of the data fusion unit.

The data fusion will have to deal with delays and signal dropouts and the different accuracies and sensor rates (for instance, typical airborne radar has azimuth and elevation accuracy of 0,2 to 2º and range accuracy of 10 to 200 feet, operating at 0,2 to 5 Hz while EO sensors have azimuth and elevation accuracy of 0,1 to 0,5º and it operates at 20 Hz [12]) Besides, they have to take into account the different coordinate frames (including the GPS uncertainties when transforming the tracks from one frame to another) and perform the track association.

Different algorithms to perform data fusion are going to be presented in the next sections, and each one has different characteristics. For SAA applications, several things should be taken into account. The first is the computational cost since the resources onboard the UAV are limited. Second, the non-linear idiosyncrasy of the application. And third, the information stored in the object's state, especially important to build the decision model. Therefore, the decision in which algorithms to use must take into account to fulfill these conditions. A suitable option could be to use K-means for data association, extended Kalman filter for state estimation and fuzzy logic for the decision. Nevertheless, a review of different algorithms will be done to give the reader a better understanding.

4.2 Classification

The multidisciplinary nature of the data fusion makes difficult to come up with clear and strict classification. Depending on the application, the data fusion techniques can be divided (1) attending to the relations between the input data sources [13], (2) according to the input/output data types and their nature [14], (3) following an abstraction level of employed data [15], (4) based on the different data fusion levels defined by the Joint Directors of Laboratories and (5) depending on the architecture type. Nevertheless, for a SAA application, the classification mentioned in the literature is the last one: depending on the architecture type, that was previously introduced in Section 2.4. It concerns where the data fusion

process will be performed and, following this criterion, the classification is made as follows:

- **Distributed architecture or sensor-level fusion**. Measurements of each source node are processed independently before the information is sent to the fusion node. Therefore, each node provides an estimate of the object's state based only on its local views, and this information is the input to the fusion process, which provides a fused global view. In this configuration, the computational power has to be available at the sensor level. In addition, from the point of view of estimating the kinematic state of the target, the sensor-level tracks, and not the measurements, are fused. Track to track fusion is a complex task and requires a certain amount of computational expense [16]. Sensor-level fusion reduces the exchange of data between nodes, paying the cost of a higher computational load.

- **Centralized architecture or central-level fusion**. The fusion node resides in the central processor that receives information from all input sources. Therefore, all fusion processes run on a central processor that uses the raw measurements provided from the sources, which are combined to obtain a single set of tracks. If it is assumed that data alignment and data association are performed correctly and that the time required to transfer the data is not significant, then the centralized scheme is theoretically optimal, but that assumption is not true for real systems. Besides, the large amount of bandwidth required to send raw data over the network is another disadvantage of the centralized approach. This problem becomes a bottleneck when this type of architecture is used to fuse data into visual sensor networks. Finally, delays in information transfer between different sources are variable and affect to the centralized scheme results to a greater extent than in other schemes. But central-level fusion major drawback is that if a sensor measurement is degraded, it affects the entire estimation process. Globally, the computational load is lower than in sensor-level architecture, however, generally more data have to be exchanged between network nodes.

These architecture were already represented in Figures 2.5(b) and 2.5(c) respectively.

One can also think about hybrid fusion configurations which try to combine the advantages of centralized and distributed architectures by supposing that both raw sensor data and sensor-level tracks can be combined in the fusion processor, but the disadvantage is in terms of increased processing complexity and possibly increased data transmission rates. In practice, there is no single optimal architecture, and the selection of the most appropriate architecture should be made based on requirements, demand, existing networks, data availability, node processing capabilities and the organization of the data fusion system.

4.3 Data Association

When talking about data fusion, the following problems could arise in the reader's mind:

- Each sensor's observation is received in the fusion node at discrete time intervals.

- The sensor might not provide observations at a specific interval.
- Observations could be noise.
- The observation generated by a specific target in every time interval is not known (a priori).

The data association techniques aim to determine from a set of measurements which ones correspond to each target. The data association is often carried out before the estimation of the state of the detected targets and it is a key step because the estimation or classification will behave incorrectly if the data association phase does not work consistently. However, this process could appear in all of the fusion levels but, for a SAA system, is in the sensor processing where it takes special importance because, roughly speaking, it will determine the number of obstacles that a sensor is sensing.

The following algorithms give a good performance for the data association task:

- K-means
- Probabilistic Data Association (PDA)
- Joint Probabilistic Data Association (JPDA)
- Multiple Hypothesis Test (MHT)
- Distributed Joint Probabilistic Data Association
- Distributed Multiple Hypothesis Test
- Graphical Models

4.3.1 K-Means

K-means is an algorithm that divides the values of the dataset into K different clusters, attending how similar the values are. It is an iterative algorithm that, from the dataset and the number of desired clusters K as an input, obtains the centroid of each cluster by

1. randomly assigning the centroid of each cluster,
2. matching each data point with the centroid of each cluster and
3. moving the cluster centers to the centroid of the cluster.

If the algorithm does not converge, it returns to the step (2). It is a well-known algorithm and is relatively easy to implement. Also, it can find a feasible solution (not optimal) in a short time. Nevertheless, the major drawback for a SAA application where the environment is constantly changing is that it needs to know the number of clusters beforehand and this number must be the optimum to provide good results. Some options to overcome it are to run the algorithm several times with different values of K until an adequate result is obtained (a solution with less variance is also a way to try to find the most optimal

solution) or to obtain the number of clusters with another technique (e.g. nearest neighbours algorithm in a non-cluttered environment).

4.3.2 Probabilistic Data Association

This algorithm was proposed in Bar-Shalom and Tse [17] to deal with a cluttered environment. It assigns an association probability to each hypothesis from a valid measurement, i.e. an observation that falls in a validation gate of a target in a given time instant. The state estimation of the final target is computed as a weighted sum of the estimated state under all the hypothesis. However, it has three main disadvantages:

- Poor performance when the targets are close to each other or crossed.
- PDA can not deal properly with multiple targets because the false alarm model does not work well.
- It will lose a target if it makes abrupt changes in its movement patterns, which is the case of UAS SAA applications.

4.3.3 Joint Probabilistic Data Association

This algorithm is a suboptimal version of the PDA able to deal with multiple targets. The difference lies in that the association probabilities are computed using all the observations and targets, i.e. it considers various hypothesis together and combines them. The main restrictions of JPDA are that a measurement only can come from one target and at one time instant, two measurements can not be originated by the same target. Note that the sum of the probabilities assigned to one target must be 1. Therefore, for a known number of targets, it evaluates the different options of the measurement-target association (for the most recent set of measurements) and combines them into the corresponding state estimation.

Regarding to the advantages and disadvantages, on one hand, it obtains better results than other algorithms (such as the MHT, explained in the next section) in situations with a high density of false measurements but, on the other hand, it is computationally expensive since the number of hypothesis increases exponentially with the number of targets.

4.3.4 Distributed Joint Probabilistic Data Association

This distributed version of the JPDA algorithm was proposed by Chang et al. [18]. However, the equations used to estimate the state of the target using several sensors assume that communication exists after every observation. An approximation can be made in case of sporadic communication and substantial amount of noise, but this algorithm remains as a theoretical model and needs to be developed to overcome the limitations for practical applications.

4.3.5 Multiple Hypothesis Test

Using only two consecutive observations to make the association could lead to an error. Then, the idea behind the MHT algorithm is using more observations in order to have a lower probability of getting an error. To do that, MHT estimates all the possible hypotheses and maintains new ones in each iteration. It was created to track multiple targets in cluttered environments, thus, it becomes an estimation technique as well. To calculate the hypotheses, Bayes rule or Bayesian networks are used. The algorithm by Reid [19] is considered the standard MHT.

Each iteration of this algorithm starts with a set of correspondence hypotheses. Each hypothesis is a collection of disjointed tracks, and the prediction of the target at the next instant is calculated for each hypothesis. The predictions are then compared with the new observations using a distance metric. The set of associations established in each hypothesis (based on distance) introduces new hypotheses in the next iteration, which represent new sets of targets based on current observations. Each new measurement can come from a new target in the field of vision, a target being tracked, or noise in the measurement process. It is also possible that a measurement is not assigned to a target because the target disappears, or because it is not possible to obtain a measurement of the target at that time. MHT also can detect a new track while maintaining the hypothesis tree structure. The probability of a true track is given by the Bayes decision model, as said before. The algorithm considers all possibilities, including both track maintenance and track initialization and removal in an integrated framework. MHT computes the possibility of having an object after generating a set of measurements using a comprehensive approach, and the algorithm does not assume a fixed number of targets. The major challenge is the effective hypothesis management.

MHT is better than other algorithms such as JPDA for the lower densities of false positives. Nevertheless, the computational cost of the MHT increases exponentially when the number of tracks, measurements or false positive is incremented. For this reason, the practical implementation of this algorithm is limited due to the cost is exponential in both time and memory. To reduce the computational cost, several approaches have been presented. Streit and E. Luginbuhl [20] presented a probabilistic MHT algorithm in which associations are considered random variables that are statistically independent and in which an exhaustive search enumeration is avoided and, for that, it is assumed that the number of targets and measurements is known. Cox and Hingorani [21] presented an efficient implementation of MHT where the best set of k hypotheses is determined in polynomial time to track the points of interest. It was the first version used to perform tracking in visual environments.

Other question is that MHT only employs one characteristic (typically position) to perform the track. Liggins et al. [22] used a Bayesian combination to overcome that and to use multiple characteristics.

4.3.6 Distributed Multiple Hypothesis Test

MHT algorithm also has a distributed version, as JPDA had it, however, the computational cost of this algorithm is even higher than the MHT, thus, its implementation in practical applications is even more

difficult.

4.3.7 Graphical Models

An explicit representation of the full joint is intractable since it is computationally expensive to manipulate and the distribution is cognitively impossible to obtain by human experts and statistically unfeasible to learn from data due to the enormous amount of parameters to estimate. A graph, where the nodes are the variables, the edges the relationships between variables and the plates the replication of a substructure, with the appropriate indexing of the relevant variables, represents the joint probability distribution among the variables, i.e. all their probabilistic dependencies. Dependencies reduce the parameters to estimate. The main graphical models are Bayesian networks (directed graph) and Markov random fields (undirected). Directed refers to the nodes can have parents or child, while in undirected graphs the nodes are just neighbours. On one hand, directed graphs are useful to express casual relationships between the variables. For instance, in the Bayesian networks each node X has a conditional probability distribution $P(X|Pa_X$, where $Pa_X)$ denotes the set of all parents of the node X, and the topology of the network specifies local conditional independencies $(X \perp NonDescendants_X|Pa_X)$ as well. On the other hand, undirected graphs are better suited to express soft constraints between the variables.

The complexity of this kind of methods is reasonable and less than the complexity of the MHT family algorithms, but special attention is needed for the correlated variables when building the graphical model.

4.4 State Estimation

From a set of redundant observations for one target, the goal is to find the set of parameters that provides the best fit to the observed data. It is also task of these algorithms to determine the state of the target (such as position) normally under movement for the SAA applications, given the different observations of each sensor. Thus, these techniques can fall under tracking techniques. The state estimation phase is a common stage in data fusion algorithms because the target's observation can come from different sensors, and the ultimate goal is to obtain a global target state from the observations. The estimation problem involves finding vector state values (e.g., position, velocity, and size) that fit the observed data as closely as possible. In general, these observations are corrupted by errors and noise propagation in the measurement process, so it has to be taken into account in the algorithms that will perform this task.

State estimation methods could be divided into two groups:

- Linear dynamics, when the equations of the object state and the measurements are linear, the noise follows a normal distribution and the environment is not cluttered. The estimation problem has a standard solution and the optimal theoretical solution is based on the Kalman filter.
- Non-linear dynamics, when the estimation problem becomes difficult. There is not an analytical solution to solve it in a general manner. It is the case of the UAS SAA problem and some variations

of the Kalman filter can deal with non-linearities such as the extended Kalman filter.

Most state estimation methods are based on control theory and use probability laws to calculate a vector state from a vector measurement or a stream of vector measurements. The next methods are used for state estimation:

- Maximum Likelihood (ML) and Maximum Posterior (MAP)
- Kalman Filter (KF), Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF)
- Particle Filter (PF)
- Distributed Kalman Filter
- Distributed Particle Filter
- Covariance Consistency Methods (Covariance Intersection and Covariance Union)

4.4.1 Maximum Likelihood and Maximum Posterior

ML is an estimation method based on probabilistic theory. Given x the state being estimated, a fixed but unknown point from the parameter space, and $z = (z(1), ..., z(k))$ the sequence of k previous observations of x, $\lambda(x)$ is the likelihood function defined as a probability density function of the sequence of z observations given the true value of state x.

$$\lambda(x) = p(z|x) \tag{4.1}$$

ML finds the value of x that maximizes the likelihood function.

$$\hat{x}(k) = arg\,\max_{x} p(z|x) \tag{4.2}$$

The hat on the top of a variable uses to mean that it is an estimation. This notation is widely used in this kind of algorithms.

Probabilistic estimation methods are appropriate when the state variable follows an unknown probability distribution. But the main disadvantage of ML is that it requires to know the empirical model of the sensor to provide the prior distribution and to compute the likelihood function. It also underestimates the variance of the distribution systematically, which could lead to a bias problem, although the bias of the ML solution becomes less significant as the number N of data points increases and it is equal to the true variance of the distribution that generates the data at $N \to \infty$.

MAP is used when x is the output of a random variable with a known probability density function $p(x)$. MAP finds the value of x that maximizes the posterior probability distribution.

$$\hat{x}(k) = arg\,\max_{x} p(x|z) \tag{4.3}$$

Both methods are equivalent when there is no a priori information on x, i.e. when there are only observations.

4.4.2 Kalman Filter

KF is the most well-known and popular estimation technique. Proposed by Kalman [23], it has a lot of application in many different fields of engineering. If the system could be described as a linear model and the error could be modeled as the Gaussian noise, then the recursive KF obtains optimal statistical estimations [24]. But when dealing with an UAS SAA application the system is not linear and other methods are required to address nonlinear dynamic models and nonlinear measurements. Here it comes the EKF, which is an optimal approach for implementing nonlinear recursive filters [25] widely used for fusing data in robotics, and the UKF.

The differences between each filter are shown in the next table of equations 4.4.

KF	EKF	UKF
	$\boxed{\mathbf{F} = \left.\frac{\partial f(\mathbf{x}_t, \mathbf{u}_t)}{\partial \mathbf{x}}\right\|_{\mathbf{x}_t, \mathbf{u}_t}}$	$\boldsymbol{\mathcal{Y}} = f(\boldsymbol{\chi})$
$\bar{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{u}$	$\boxed{\bar{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})}$	$\bar{\mathbf{x}} = \sum w^m \boldsymbol{\mathcal{Y}}$
$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^\mathsf{T} + \mathbf{Q}$	$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^\mathsf{T} + \mathbf{Q}$	$\bar{\mathbf{P}} = \sum w^c (\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})^\mathsf{T} + \mathbf{Q}$
		$\boldsymbol{\mathcal{Z}} = h(\boldsymbol{\mathcal{Y}})$
	$\boxed{\mathbf{H} = \left.\frac{\partial h(\bar{\mathbf{x}}_t)}{\partial \bar{\mathbf{x}}}\right\|_{\bar{\mathbf{x}}_t}}$	$\boldsymbol{\mu}_z = \sum w^m \boldsymbol{\mathcal{Z}}$
$\mathbf{y} = \mathbf{z} - \mathbf{H}\bar{\mathbf{x}}$	$\mathbf{y} = \mathbf{z} - \boxed{h(\bar{x})}$	$\mathbf{y} = \mathbf{z} - \boldsymbol{\mu}_z$
$\mathbf{S} = \mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} + \mathbf{R}$	$\mathbf{S} = \mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} + \mathbf{R}$	$\mathbf{P}_z = \sum w^c (\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)(\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)^\mathsf{T} + \mathbf{R}$
$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}\mathbf{S}^{-1}$	$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}\mathbf{S}^{-1}$	$\mathbf{K} = \left[\sum w^c (\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)^\mathsf{T}\right] \mathbf{P}_z^{-1}$
$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}$	$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}$	$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}\mathbf{y}$
$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}$	$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}$	$\mathbf{P} = \bar{\mathbf{P}} - \mathbf{K}\mathbf{P}_\mathbf{z}\mathbf{K}^\mathsf{T}$

(4.4)

EKF is used in many literature such as Fasano et al. [26] to perform this step of the data fusion in SAA applications (its differences with the linear KF have been highlighted in boxes). Nevertheless, the computation of the Jacobians are extremely expensive and attempts to reduce it, such as linearization, can lead to errors in the filter and instability. On the other hand, the UKF does not have the linearization step and the associated errors of the EKF [27]. In Labbe [28] it is said that the UKF is more accurate and typically more stable, in general, not particularly for the application.

4.4.3 Distributed Kalman Filter

Described in [22], this variation used in distributed architectures has to take two main things into account. The algorithm would require the synchronization of the clocks of each source [29]. The synchronization is typically achieved by using protocols that employ a shared global clock, such as the network time protocol. Otherwise, the estimation would be quite inaccurate [30]. But if the estimations are consistent and the cross-covariance is known and determined exactly (or the estimations are uncorrelated), then it is possible to use the distributed KF [31].

4.4.4 Particle Filter

Particle filters are recursive implementations of the Monte Carlo sequential methods [32]. This method builds the posterior density function using several random samples that are the so-called particles. Particles are propagated over time with a combination of sampling and resampling steps. In each iteration, the sampling step is used to discard some particles, increasing the relevance of the regions most likely to follow. In the filtering process, several particles of the same state variable are used, and each of them has an associated weight indicating the quality of the particle. The estimation will be the result of the weighted sum of all particles.

Therefore, there are two main steps in the PF algorithm: predicting and updating. In the first one, each particle is modified according to the existing model and noise at a time instant and, in the second one, the weight of each particle is reevaluated using the last sensor observation so the particles with lower weights are removed. More in depth, the algorithm comprises the following tasks:

1. Initialization of the particles
2. Prediction step
3. Evaluate the particle weight
4. Select the particles with higher weights while removing those with lower weights and update the state with the new particles
5. Propagate the result for the next time instant

This algorithm is more flexible than KF and it can deal with non-linearities as well as the EKF. It also can face non-Gaussian densities in the dynamic model and in the noise error. Nevertheless, the drawback resides in the large number of particles needed to obtain small variance in the estimator. In addition, it is difficult to establish the optimal number of particles beforehand which has an impact on the computational time since a great number of particles means a higher cost. There has been some research [33] in using a dynamic number of particles instead of a fixed number as in earlier versions of the algorithm.

4.4.5 Distributed Particle Filter

This algorithm, in the context of sensor fusion, tries to solve out-of-sequence measurements by regenerating the probability density function to the time instant of those measurements [34]. This uses to take a lot of computational cost and needs a lot of space to store previous particles. Orton and Marrs [35] proposed to store the information on the particles at each time to save the cost of recalculating that information. Close to optimal, when the delay increases the effect on the result is minimal [36]. The large amount of space to store the state of the particles at each time instant remains a problem.

4.4.6 Covariance Consistency Methods

These methods are used in distributed networks. Knowing cross variances is not a constraint in contrast to distributed KF. They were proposed by Uhlmann [31] and they are general and fault-tolerant frameworks for maintaining covariance means and estimations. There are two main methods: covariance intersection and covariance union.

Covariance Intersection

This method was born to compute the cross variance X, that should be known exactly for a good performance of the KF, in a computational time friendly way. Given two estimations to combine, (a, A) and (b, B) ((mean, covariance)), a joint covariance M is defined as

$$M \geq \begin{bmatrix} A & X \\ X^T & B \end{bmatrix} \tag{4.5}$$

The covariance intersection algorithm computes this joint covariance matrix M that could be used in the KF equations that would provide the best fused estimation (c, C). The purpose of this method is to provide a fused estimation with a lower associated uncertainty by generating the covariance matrix M. But what it does differently from the KF? In the case of two estimations with equal covariance $A = B$, KF would assume statistical independence and it would provide a fused estimation with covariance $C = (1/2)A$. Covariance intersection does not assume this independence, being consistent even if the estimations are completely correlated, and it will provide a fused covariance $C = A$. In case $A \leq B$, covariance intersection does not provide information about the estimation (b, B) and the fused estimation would be (a, A).

The use of the covariance intersection algorithm guarantees consistency and non-divergence. Consistency is guaranteed because every joint consistent covariance is enough to produce a fused estimation, while choosing a measurement such as the determinant, minimized in each fusion operation, provides a non-divergence criterion since the size of the estimated covariance would not be increased according to this criterion. Nevertheless, the algorithm does not work properly when the measurements to be fused are inconsistent (different estimations with high accuracy and small variance but a large difference from the states of the others).

Covariance Union

Covariance union is proposed to solve the problem of inconsistent measurements. This problem arises when the difference between the average estimations is greater than the covariance provided.

Inconsistent inputs can be detected when the Mahalanobis distance [37] between them, given in Equation 4.6, is larger than a given threshold.

$$M_d = (a - b)^T (A + B)^{-1} (a - b) \tag{4.6}$$

A high Mahalanobis distance could indicate that the estimations are inconsistent, but the threshold needs to be established by the user or learned automatically.

The covariance algorithm works as follows. Provided two observations, it is known that one of them is correct and the other is not, but not which is each one. The observation needs to be updated with a measurement consistent with both or the estimations for the KF to provide a consistent solution. Since there is no way to determine which estimation is correct, it is necessary to provide a new estimation (u, U) that is consistent with both estimations, following the next properties

$$\begin{aligned} U &\geq A + (u-a)(u-A)^T \\ U &\geq B + (u-b)(u-B)^T \end{aligned} \tag{4.7}$$

where some measurement of U is minimized. One strategy is to assign the value of one measurement to the mean of the new estimation, e.g. $u = a$, and then to chose the value of U that makes the estimation consistent with the case in which the other measurement b was the correct. It is also possible to assign to u a value between the two measurements a and b with the purpose to decrease the value of U. Convex optimization algorithms must be employed to solve the inequalities in the Equations 4.7 in order to obtain U, for instance, the iterative method described in [38].

After all, it is obtained a fused value u with the less covariance possible that makes the two measurements consistent. If the obtained covariance is much larger than the initial ones, it reveals uncertainty between the initial estimations. Note that this method is also easily extensible to more inputs.

4.5 Fusion Decision

These techniques aim to make a high-level inference about the events and activities that are produced from the detected targets. These techniques use the knowledge of the perceived situation, provided by many sources and in different ways depending on the data fusion application. It is the fusion itself. The fusion process requires reasoning while taking into account the uncertainties and constraints of the system. Algorithms used in this field to perform the decision-making are:

- Bayesian Methods
- Dempster-Shafer Inference
- Abductive Reasoning and Fuzzy Logics
- Semantic Methods

4.5.1 Bayesian Methods

The Bayesian inference is based on the Bayes rule

$$P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)} \tag{4.8}$$

where the posterior probability $P(Y|X)$ represents the belief in the hypothesis Y given the information X. It is a way to combine the data according to the probability theory rules, where uncertainties are represented with the conditional probability terms describing beliefs on this data (from 0 to 1, 0 would represent lack of belief and 1 total belief). The major drawback of the Bayesian inference is that the probabilities $P(X)$ and $P(X|Y)$ must be known or estimated (using Bayesian programming for the conditional probabilities [39], for instance). In the *Handbook of Multisensor Data Fusion* [40], the following problems are described regarding the Bayesian inference.

- Difficulty in establishing the value of a priori probabilities.
- Complexity when there are multiple potential hypotheses and a substantial number of events that depend on the conditions.
- Hypothesis should be mutually exclusive.
- Difficulty in describing the uncertainty of the decisions.

4.5.2 Dempster-Shafer Inference

The Dempster-Shafer theory provides a formalism used to represent incomplete knowledge and combination of evidence and allows to represent uncertainty explicitly [41]. The mathematics of this method can be found in both in Dempster [42] and Shafer [43].

With this method, a priori probabilities are not required because they are assigned at the instant the information is provided, not like in the case of the Bayesian method. Some uses of it can be founded in [44] and its extension [45], where information is fused in context-aware environments dynamically modifying the associated weights to the sensor measurements, thus, it lets to calibrate the fusion mechanism according to the recent sensor measurements.

4.5.3 Abductive Reasoning and Fuzzy Logics

The abduction method attempts to find the best explanation for an observed event. Thus, it is more a reasoning pattern than a data fusion technique, for that reason it needs to be complemented with a different inference method. It is then when fuzzy logic comes. Fuzzy logic provides an ideal tool for inexact reasoning, particularly in rule-based systems and it has had some notable success in practical application. It has been used in sensor fusion applications, e.g. for LIDAR and camera fusion in Zhao et al. [46], where the following advantages of fuzzy logic are highlighted:

- It is built on top of the knowledge and experience of experts thus it can employ not only results from the sensors but also a priori knowledge.
- It can model nonlinear functions of arbitrary complexity.

- It can tolerate imprecise results of two sensors.
- It is a flexible fusion framework, i.e., more sensors can be easily integrated into the system in the future.

It works as follows. Given a universal set $\mathcal{X}$ with elements x and a subset $\mathcal{A}$ whit elements x that have some properties, there is a membership function μ that assigns a value between 0 and 1 indicating the degree of membership of every x to the subset $\mathcal{A}$. Mathematically,

$$\begin{aligned} \{\mathcal{X}\} =& \{x\} \\ \{\mathcal{A}\} = \{x \,|\, x \quad & has\ some\ specific\ property\} \\ \mathcal{A} \rightleftharpoons & \mu_A(x) \rightarrow [0,1] \end{aligned} \tag{4.9}$$

The fuzzy membership function needs to be quantified in some way. Figure 4.2, extracted from [47], shows an example where $\mathcal{X}$ is a set with all aircraft and $\mathcal{A}$ is a set with fast aircraft.

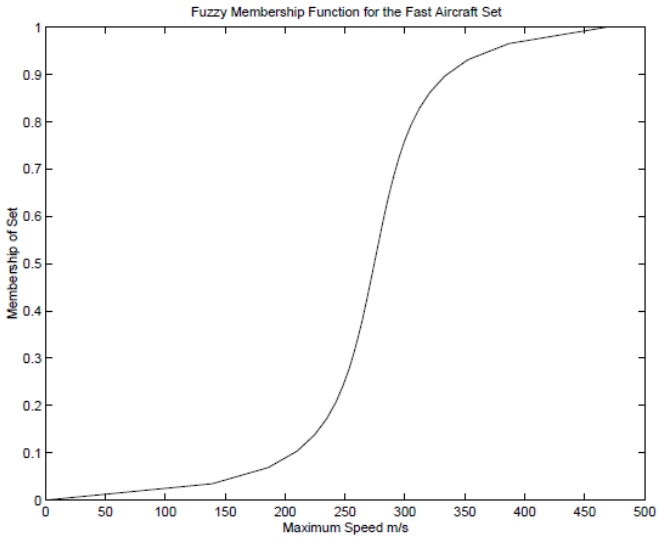


Figure 4.2: Example fuzzy membership function [47]

In the context of fuzzy logics, the operation *AND* is implemented as a minimum, *OR* as a maximum and *NOT* as a compliment.

$$\begin{aligned} \mathcal{A} \cap \mathcal{B} \rightleftharpoons & \mu_{\mathcal{A}\cap\mathcal{B}}(x) = min[\mu_{\mathcal{A}}(x), \mu_B(x)] \\ \mathcal{A} \cup \mathcal{B} \rightleftharpoons & \mu_{\mathcal{A}\cup\mathcal{B}}(x) = max[\mu_{\mathcal{A}}(x), \mu_B(x)] \\ \overline{\mathcal{B}} \rightleftharpoons & \mu_{\overline{\mathcal{B}}}(x) = 1 - \mu_{\mathcal{B}}(x) \end{aligned} \tag{4.10}$$

Also, the properties associated with binary logic are held (commutativity, associativity, idempotence, distributivity De Morgan's law and absorption). The only exception is the law of the excluded middle,

which is no longer true.

$$\begin{aligned} \mathcal{A} \cup \overline{\mathcal{A}} &\neq \mathcal{X} \\ \mathcal{A} \cap \overline{\mathcal{A}} &\neq \emptyset \end{aligned} \tag{4.11}$$

Thus, these definitions and laws provide a systematic way to reason about inexact values.

4.5.4 Semantic Methods

Semantic data models are models based on relationships between stored symbols and the real world. In a semantic scheme, the outputs of the nodes that process the raw data are only the semantic information. Two phases are needed, one offline, that incorporates the most appropriate knowledge into semantic information, and a second one in real time, in which relevant attributes are fused providing a semantic interpretation of the sensor data. Another way to see it is that the data obtained from the environment by the sensor is translated into a formal language, which is compared with similar languages stored in the database.

Therefore, the nodes do not need to transmit raw data but a formal language structure, which provides savings in the cost of transmission. Nevertheless, the set of behaviours must be stored in advance and it might be difficult in some scenarios.

4.6 Other solutions

Other way to use the information of two or more sensors to enhance the performance of a single one is presented in the schema in Figure 4.3.

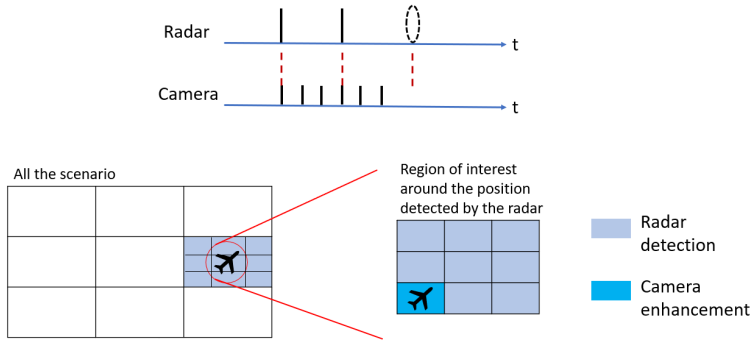


Figure 4.3: Schema of a camara-based enhancement of the radar measurement

It is proposed a radar as a primary source of information. The radar has as an advantage among LIDARs a longer detection range, nevertheless, it has a low resolution, i.e. in a given scenario the output *pixel* is very large. As a secondary source, a camera. A camera is also a long range solution with high

resolution but it is more likely to produce a false detection due to a cluttered environment. Combining both sensors like in Figure 4.3 helps to overcome the limitations of each sensor.

To avoid getting a false detection from the camera, its information is only employed when a measurement from the radar is received. However, this radar measurement would not be as accurate as desirable, then the camera helps to enhance it. Instead of running the algorithm for the camera obstacle detection over all the scenario, it is run only in a region of interest created around the position measured by the radar, reducing considerably the chances of a false alarm. Therefore, an accurate position of the target will be obtained by the camera.

It was found that this was already used in some papers such as [48], thus it has been validated to work.

Chapter 5

Machine and Deep Learning for obstacle detection using visual electro-optical sensors

When one looks at the sensors used in a SAA system, it is quickly identified that all the architectures have almost a camera in their configurations, both as a secondary source of information or as a primary source in novel systems. Thus, this chapter is dedicated to describe how to proceed in the field of image processing for obstacle detection in a SAA application using machine and deep learning techniques. The application is centered in evaluating if there is an obstacle in the clear sky, above the horizon.

Section 5.1 introduces the artificial intelligence to the reader while Section 5.2 and Section 5.3 explain the basis of the techniques that are used, Convolutional Neural Networks and Support Vector Machine respectively. Section 5.4 discusses the implementation of a Faster Region with Convolutional Neural Networks for aircraft detection while Section 5.5 discusses the implementation of the Support Vector Machine that together with an edge detection algorithm will lead to the obstacle detection. Finally, Section 5.6 shows the results of the implementations.

5.1 Artificial Intelligence, Machine Learning and Deep Learning

Artificial intelligence (AI) was born with the idea to build complex systems with the human intelligence characteristics. The idea of the first researchers in the field of AI was to build a general AI in which machines and human beings have the same cognitive capacities, very ambitious and not reached yet. But there is also another group, the narrow AI, where the current research on AI is. Here is where the use, that is made through algorithms and guided learning with machine learning and deep learning, comes in. Nowadays, the world is living a tremendous advance in the use of machine learning and,

since a few years ago, specifically deep learning, both included in the context of AI, as it has been said.

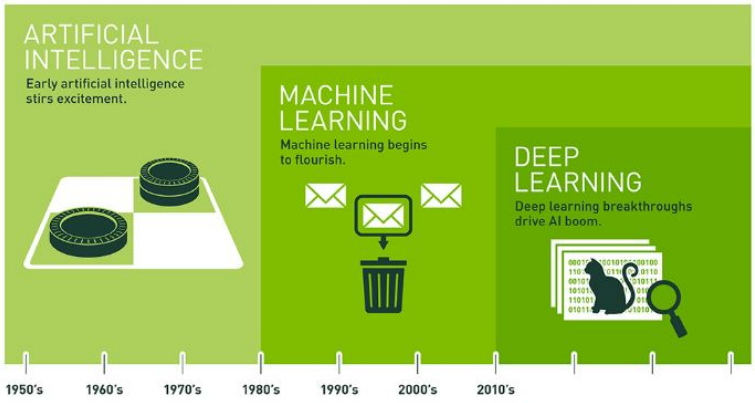


Figure 5.1: Artificial intelligence, machine learning and deep learning [49]

Machine learning, in its most basic use, is the practice of using algorithms to parse data, learn from them and then be able to make a prediction or suggestion about something. The machine is trained using a large amount of data allowing the algorithms to be perfected, instead of hand-coding software routines with a specific set of instructions to accomplish a particular task. Regarding image processing, **computer vision** has been one of the very best application areas for machine learning for many years, where a well-trained machine could classify an image attending to some specific features.

Following the evolution of machine learning in the last decade, a specific machine learning technique known as **deep learning** has spread more strongly. By definition, deep learning is a subset within the field of machine learning, which preaches with the idea of learning from example. In deep learning, instead of teaching a computer a huge list of rules to solve a problem, it is given a model that can evaluate examples and a small collection of instructions to modify the model when errors occur. Over time, it is expected that these models will be able to solve the problem extremely accurately, thanks to the system's ability to extract patterns. Although there are different techniques to implement deep learning, one of the most common is to simulate a system of **neuronal networks**. Saving the distances, they are inspired by the biological functioning of the human brain composed by the interconnection between neurons. In this simplified case, the neuronal network is composed of different layers, connections and a direction in which data is propagated through each layer with a specific analysis task. It is about providing enough data to the neuron layers so that they can recognize patterns, classify and categorize them. In the image processing field, an image can be taken as an input to the first layer. There, it will be divided into thousands of pieces that each neuron will analyze separately. It can analyze the color, the shape, etc. Each layer is an expert in a characteristic and assigns it a weight. Finally, the final layer of neurons collects that information and gives a result. Each neuron assigns a weight to the input, as a correct or incorrect result relative to its task. The output will be determined by the sum of these weights.

Table 5.1 presents some of the difference in using machine learning or deep learning.

	Machine Learning	Deep Learning
Data dependencies	Excellent performances on a small/medium dataset	Excellent performance on a big dataset
Hardware dependencies	Works on a low-end machine	Requires powerful machine, with GPU
Feature engineering	Need to understand the features that represent the data	No need to understand the best feature that represents the data
Interpretability	Some algorithms easy (logistic, decision tree) other almost impossible (SVM, XGBoost)	Difficult to impossible
Training dataset	Small	Large
Choose features	Yes	No
Number of algorithms	Many	Few
Training time	Short	Long

Table 5.1: Machine learning vs. deep learning

There are two problems when processing an image in this context: classification and detection. Classification algorithms assign a class to each pixel based on their properties. Object detection algorithms not only classifies the input image but also draw bounding boxes around the object of interest to locate it within the image.

In this all context and attending to the research being developed at CfAR, it is very interesting to use machine learning or deep learning to process images captured by a camera, because, in contrast with classical programming, it is not needed to establish all the clauses and threshold to classify the regions of the image. Instead, the trained intelligent algorithm can detect the class from its knowledge, learning which are these thresholds that lead to the classification, and also it is able to predict cases that the programmer could have overlooked. Thus, two techniques that can help with this task are explained in the next sections: convolutional neuronal networks (CNN) and support vector machine (SVM).

5.2 Convolutional Neural Networks

Within the computer vision, deep learning approaches are one of the most popular ways to analyze visual imagery. CNN or *ConvNet* is a class of deep neural networks that have been winning the Imagenet Large Scale Visual Recognition Challenge since 2012 [50]. It is difficult to understand how a CNN works: it passes the image to the network where it is sent through several convolutions and pooling layers and, finally, the output is obtained in the form of the object's class.

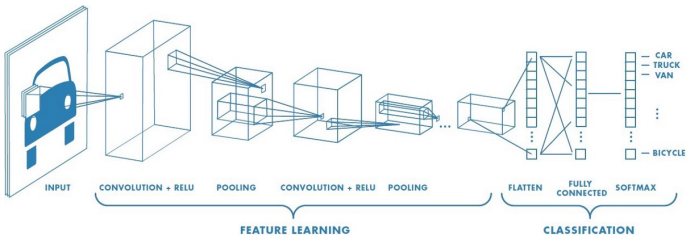


Figure 5.2: Convolutional Neural Networks [51]

There are different algorithms that could use these networks to detect objects. They are going to be explained in the next sections. The amazing thing about these techniques is that they can perform the feature extraction, the classification and the object detection all in, thus, sending the image (and the pre-trained data) to the algorithm gives the result directly.

5.2.1 Regions with CNN features

When processing an image where the number of objects of interest is unknown, to take different regions of interest and use a CNN to classify the presence of the object within that region could be a solution. But these objects can have different aspect ratios and different spatial locations, thus the number of regions needed would make it computationally unapproachable. Regions with CNN features (R-CNN) was proposed by Girshick et al. [52] to address this problem.

R-CNN combines proposals of rectangular regions with characteristics of CNN. It reduces the problem to find 2000 regions in the image that might contain an object. These regions are called region proposals and they are selected by (1) generating an initial sub-segmentation that generates candidate regions, (2) using a greedy algorithm to combine similar regions into larger ones recursively and (3) using the generated regions to produce the final candidate region proposals. In each region (reshaped), CNN acts as a feature extractor, and then these features are fed into a SVM to divide these regions into different classes. Finally, a *bounding box regression* is used to predict the bounding boxes for each identified region.

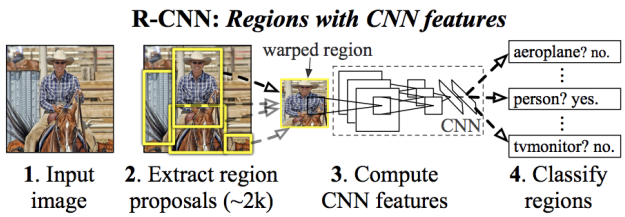


Figure 5.3: Regions with CNN features [52]

Nevertheless, it stills taking a lot of time to train the networks since it has to classify 2000 region proposals per training image. Also, the detection takes around 47 seconds [52] per test image, which

makes it not suitable for a real time application. Last, the selective search algorithm is not an intelligent algorithm thus it could lead to propose bad candidate regions.

5.2.2 Fast R-CNN

The same author proposed in [53] a solution to overcome the speed limitations of the R-CNN. Instead of feed the region proposals to the CNN, the fast R-CNN feeds the entire input image to the network to generate a convolutional feature map. It is there where the region proposals are identified. Then, they go through a region of interest pooling layer that reshapes them into a fixed size in order to pass the regions to a fully connected network that classifies them and returns the bounding boxes using *softmax* and *linear regression* layers simultaneously.

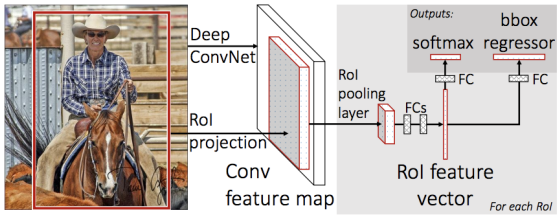
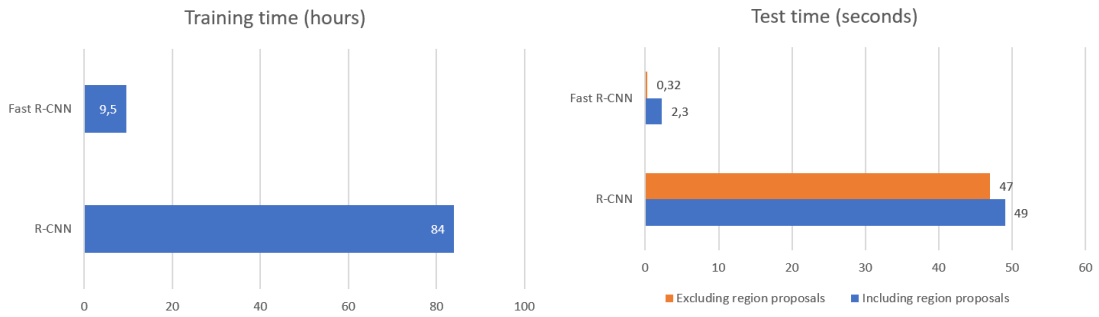


Figure 5.4: Fast R-CNN [53]

Fast R-CNN is significantly faster than R-CNN: the ConvNet is fed with one region per image instead of 2000. But the definition of the region proposals using selective search stills being a bottleneck, slowing down the process when compared to not using region proposals, as it is shown in Figure 5.5 (values extracted from [53]).

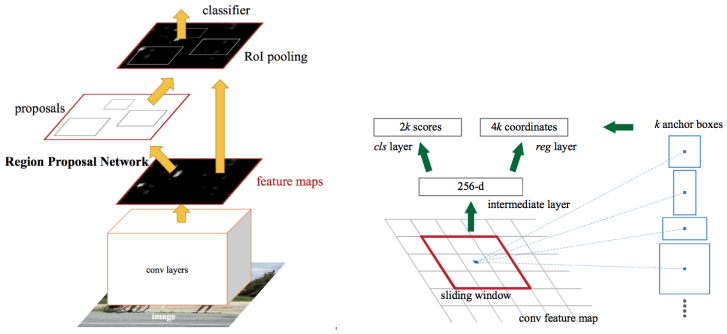


(a) (b)

Figure 5.5: Comparison of R-CNN and fast R-CNN

5.2.3 Faster R-CNN

Ren et al. [54] developed the faster R-CNN that instead of using a selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. It is called the region proposal network (RPN). The algorithm is one order of magnitude faster with this approach [54].



(a) Faster R-CNN (b) RPN

Figure 5.6: Faster R-CNN and RPN [54]

Figure 5.6(a) describes how the algorithm works while Figure 5.6(b) describes how the part of the RPN works. Faster R-CNN takes the feature maps from the CNN and passes them into the RPN, where, using a sliding window over these feature maps, it generates k anchor boxes (fixed sized boundary boxes that are placed throughout the image and have different shapes and sizes) for each window. For each anchor box, RPN predicts the probability that an anchor is an object (without considering any class) and the *bounding box regressor* is used to adjust the anchors to better fit the object. Then, the region of interest pooling layer extracts fixed size feature maps for each anchor, which are sent to a fully connected layer with *softmax* and *linear regression* layer, as well as in fast R-CNN, and finally it will come up with the classification of the object and the prediction for the bounding boxes for the identified objects.

5.3 Support Vector Machine

SVM is a well-known supervised machine learning algorithm, typically used in binary classification problems (in fact, it is used by the R-CNN to perform the classification part). When running this algorithm in an image, it can classify each pixel into the categories defined during the training, however, extra code is needed to perform the detection. It plots each data item as a point in a n-dimensional space (n being the number of features that define a class). The classification is performed by finding the hyperplane that differentiates two classes: data points falling on either side of the hyperplane can be attributed to different classes. Between all the possible hyperplanes, the algorithm would find the one that has the maximum margin (maximum distance between data points of both classes), which also provides some confidence when classifying future data points.

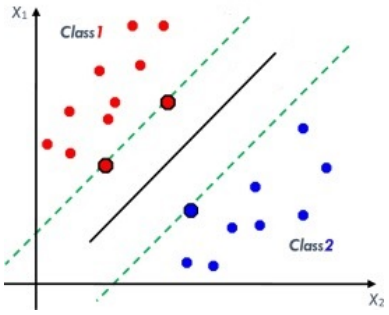


Figure 5.7: SVM classification (adapted from [55])

The support vectors are the points that are closer to the hyperplane and help to build the SVM by influencing the position and orientation of it and maximizing the margin of the classifier (the support vectors are the points highlighted in Figure 5.7).

The hyperplane depends on the number of features. For two features the hyperplane is a line, for three is a two-dimensional plane and so on. In addition, for a huge dataset, the resulting separation hyperplane is no longer that straight and classification becomes more difficult since it is not that easy to evaluate if a point is on one side or another of the hyperplane. For that reason, a trade-off between accuracy and computational time has to be made.

5.4 Implementation of Faster R-CNN for aircraft detection

Among the algorithms that use CNN for image classification, the faster R-CNN is, as it is indicated by its name, not only the faster one but also the most accurate and thus the most likely to produce good results in an onboard system. For those reasons, it is the one used. The implementation is done in Matlab. The goal with the implementation of this technique is to detect aircraft in the sensor image. For that, the algorithm has to be trained for the object class that it has to detect. Thus, in the end, it would be able to detect that class in the image. The training is done by means of the Matlab function *trainFasterRCNNObjectDetector*, which has three main inputs: *trainingData*, *network* and *options*.

To build the *trainingData*, it is needed to find a good dataset of images with whom training the network. After trying two different datasets with bad results, a third one was found. This dataset has been used in the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) VOC (Visual Object Classes) project [56], that has been running challenges to evaluate performance on object class recognition from 2005 to 2012. In the 2012 edition, the dataset contains 17125 images from 20 different classes, being *aeroplane* one of them. To each image corresponds a *XML* file where, among other information, there are the classes that contain that image and the region of the image where they are located. This is an important thing when training a CNN: it is necessary to know in which region of the training image is the class object located, since there may be other objects in the image that are not of interest. These regions are called bounding boxes. Therefore, to train a detector for aircraft, it has been created an algorithm that searches in every file for the *aeroplane* class and extracts the bounding box

information of the object. After all, it is created a table with two columns where the first column contains paths and file names to the images and the second column contains the bounding boxes related to the corresponding image. Each bounding box must be in the format [x y width height]. The format specifies the upper-left corner location and size of the object in the corresponding image. The table variable name defines the object class name. In total, there were 960 images with the class *aeroplane*.

The Faster R-CNN object detection network is composed of a feature extraction network followed by two sub-networks. The first sub-network following the feature extraction network is a RPN trained to generate object proposals (object or background). The second sub-network is trained to predict the actual class of each proposal. Matlab has some deep neural networks to use for feature extraction. It has been decided to use *ResNet-18* which makes a good trade-off between accuracy and computational time. The images sent to the detector will need to have a minimum resolution of 224x224. Results in terms of accuracy and prediction time for the different networks will be shown further in this section.

Regarding the options, they have been selected after different trainings trying to optimize both accuracy and computational time during the classification and detection. To highlight, the solver uses stochastic gradient descent with momentum optimizer, the initial learning rate is set to 10^{-3} (if the learning rate is too low, then training takes a long time but if the learning rate is too high, then training might reach a suboptimal result or diverge). The maximum number of epochs, which is the full pass of the training algorithm over the entire training set, is set to 30 in order to make the detector robust enough without making it too slow to train (it has been tested that to double this value, i.e., 60, only produces a very small difference in the results). And the mini-batch size, which is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights, is set to 1, which in turn allows for different sizes of the image in the training dataset preventing them from being batched together for processing.

Before running the training, 99 of the dataset images have been reserved for testing the detector's performance. After training, a detector file is generated which, used with the image in which it is wanted to perform the detection by means of the Matlab function *detect*, provides as an output the boxes in which the class of interest is located, in this case the aircraft, and a value of confidence in that detection. The detection threshold has been set, based on the experimentation, to 0,2, i.e., it would need a less confidence to provide a detected obstacle, since a false positive is better than a false negative in this application. The testing images have been used to determine the accuracy and computational time of each detector.

To evaluate the accuracy, the precision/recall curve highlights how precise a detector is at varying levels of recall. Precision (or positive predicted value) is the fraction of true positives *tp* (from the results identified as positive, the ones that actually are) and true positives and false positives *fp* (results identified as positives that are not). Recall (or sensitivity) is the fraction of true positives and true positives and false negatives *fn* (results identified as negatives that are positives). Ideally, the precision would be 1 at

all recall levels.

$$
\begin{aligned}
Precision &= \frac{tp}{tp+fp} = \frac{(relevant\ elements) \cap (selected\ elements)}{selected\ elements} \\
Recall &= \frac{tp}{tp+fn} = \frac{(relevant\ elements) \cap (selected\ elements)}{relevant\ elements}
\end{aligned}
\tag{5.1}
$$

Where the *relevant elements* are all the black points in the green side (not only the circle) in Figure 5.8 and *selected elements* the elements within the circle. A threshold of 0,5 has been chosen to determine the percentage of overlapping area between the ground truth data and the detection that is considered as a success.

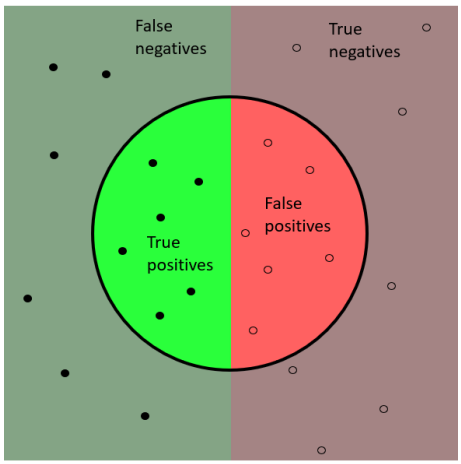
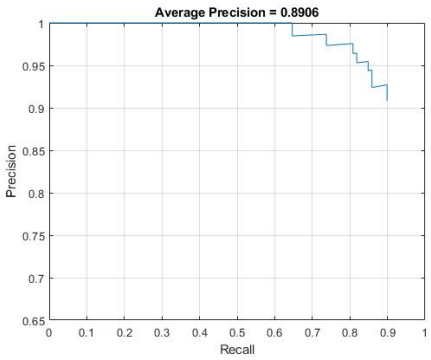


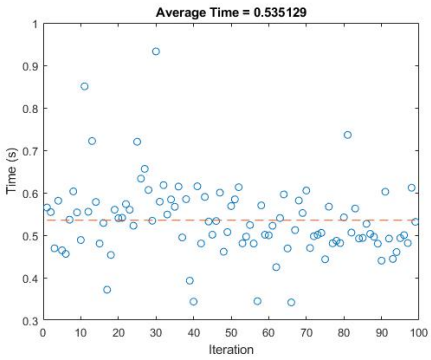
Figure 5.8: Relevant elements and selected elements

To evaluate the prediction time, the detector is run several times for each one of the test images and then the mean is computed. The characteristics of the computer where the results are obtained from are AMD Ryzen 5 2600X Six-Core (3.60 GHz) CPU, 16 GB RAM and NVIDIA Quadpro P2000 GPU. Consider that the times will be reduced if run in another language like C.

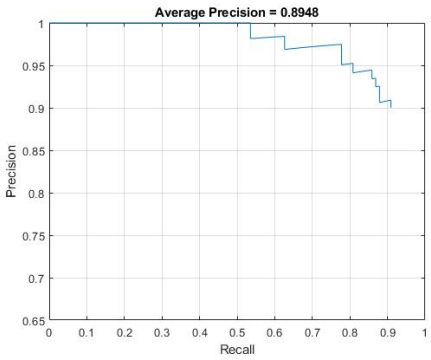
Figures 5.8 and 5.9 show the results where the precision for each network is represented in the first column and the prediction time in the second.



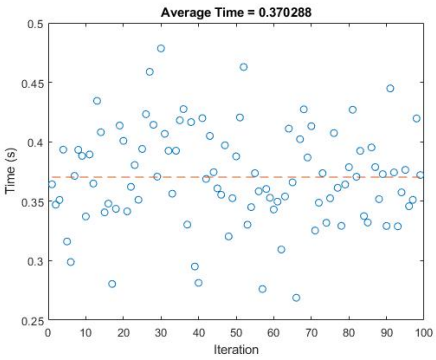
(a) GoogleNet Accuracy



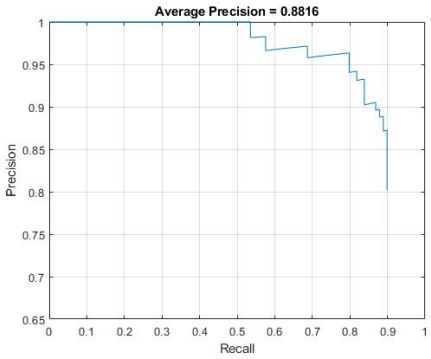
(b) GoogleNet Prediction time



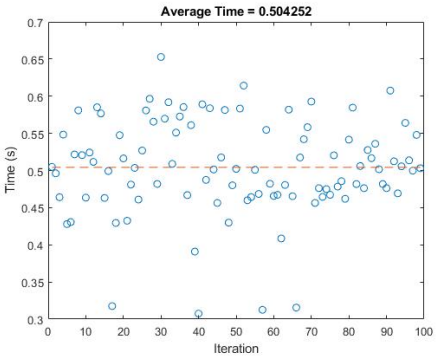
(c) ResNet-18 Accuracy



(d) ResNet-18 Prediction time

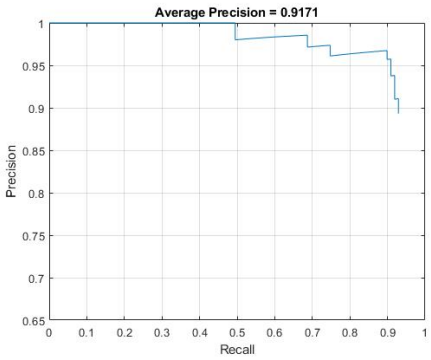


(e) MobileNet-v2 Accuracy

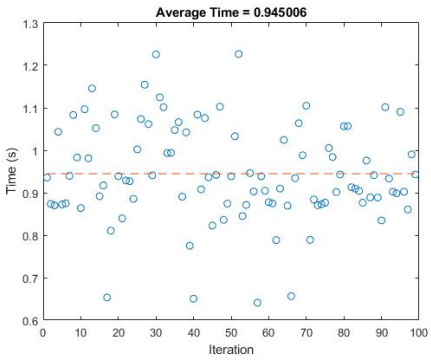


(f) MobileNet-v2 Prediction time

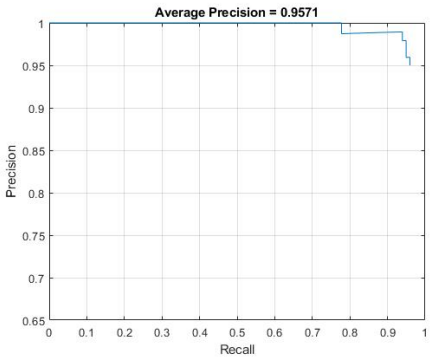
Figure 5.8: Accuracies and prediction times for different networks (I)



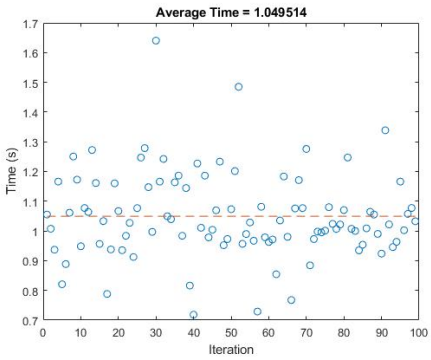
(g) ResNet-50 Accuracy



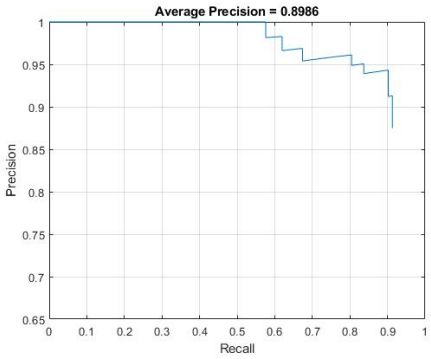
(h) ResNet-50 Prediction time



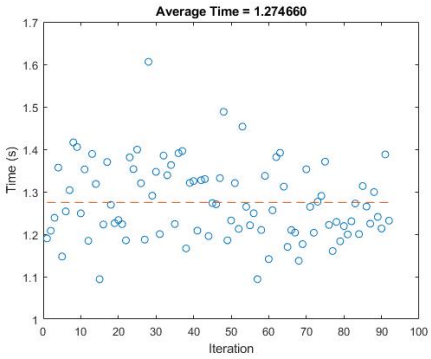
(i) ResNet-101 Accuracy



(j) ResNet-101 Prediction time



(k) Inception-v3 Accuracy



(l) Inception-v3 Prediction time

Figure 5.9: Accuracies and prediction times for different networks (II)

ResNet-18 network has the best trade-off between accuracy and time and for that reason is the one selected for the implementation, since its prediction time allows for a less time-consuming implementation, the accuracy is 6,51 % less than the one achieved with the most accurate network, nevertheless, it takes 35,28 % of its time.

Finally, the implementation of this method will return an easily interpretable array of size $numberObstacles + 4 \cdot numberObstacles$, being the first element the number of obstacles, the next four the X coordinate and Y coordinate of the top-left corner of the bounding box, the height and the width of that bounding box, respectively, and so on until reaching the total number of obstacles.

5.5 Implementation of SVM and edge detection for obstacle detection above the horizon

An edge detection algorithm is a very fast way to detect objects, nevertheless, it can suffer a lot from ground clutter and thus produce a lot of false detections. There is where the SVM comes in. The machine learning technique is used for classifying both ground and sky and then the edge detection algorithm will work only in the sky part of the image. Therefore, there are two main steps in the implementation, that has been done through a function where the input is the video frame and the output is an array containing the number of obstacles, defined as spheres, and thus their centroids and radius.

5.5.1 Sky segmentation

To separate sky and ground, a SVM is trained for classifying these two classes and then the line that best separates them, i.e. the horizon line, will be found. The idea of using SVM to detect the horizon line was introduced in [57] but with others purposes, thus, it has been adapted for the current application.

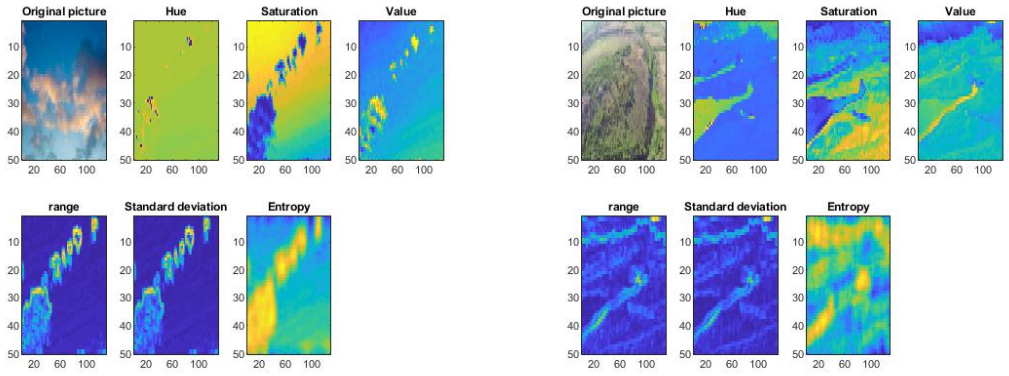
To perform the training it was created a dataset of 202 images of each class. Nevertheless, with such an amount of images, the file created after the training was too heavy which made the classification computationally expensive. Thus, the results that will be shown are built with a dataset of 29 images of each class, that provides also a good performance in the major part of the tested images and reduces the computational time one order of magnitude.

In the case of machine learning is the programmer the one who establishes the features that will lead to the classification, then, to select them is the first step of the training algorithm. The features selected are:

- Hue: value from 0 to 1 that corresponds to the color's position on a color wheel. As hue increases from 0 to 1, the color transitions from red to orange, yellow, green, cyan, blue, magenta, and finally back to red.
- Saturation: amount of hue or departure from neutral. 0 indicates a neutral shade, whereas 1 indicates maximum saturation.
- Maximum value among the red, green, and blue components of a specific color.
- Range value (maximum value to minimum value) of the 3 by 3 neighborhood around the corresponding pixel in the input image.
- Standard deviation of the 3 by 3 neighborhood around the corresponding input pixel.
- Entropy value of the 9 by 9 neighborhood around the corresponding pixel in the input image.

The feature extraction is done for both sky and ground images and then two giant matrices where each column represents one feature are built. This matrix is then sent to the Matlab function *fitcsvm*

which performs the training. The gaussian or radial basis function kernel worked the best for this problem. Also, sequential minimal optimization is used. This function returns a *struct* with the information needed to classify future pictures. This *struct* will be loaded in the first part of the implementation. Results of the training are shown in Figure 5.10.



(a) Sky

(b) Ground

Figure 5.10: Sky and ground features

In the first part of the implementation, from a video frame, a function in charge to perform the sky segmentation will return the slope m, the intercept b and the R^2 of the horizon line $y = mx+b$. Inside this function, first, the image is scaled down to save memory and computational time. The same features used in the training are extracted and through the Matlab function *predict*, that uses the trained data, the segmentation is performed. Then every pixel is classified either as 1 (ground) or as 0 (sky) and is brought back to the original picture representation obtaining a binary image of the environment. The image is smoothed with some filters and sent to the Matlab function *edge* using the *Prewitt* edge detection algorithm, that calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result shows how abruptly or smoothly the image changes at that point, and therefore how likely it is that part of the image to represent an edge, as well as the most likely orientation of that edge. Thus, the horizon line is extracted. A mask is applied to remove border pixels. After that, a function uses the extracted horizon line and tries to fit a first order polynomial into it, estimating also the accuracy by taking into account how good the line fit was. i.e. it is obtained the slope, the intercept and the R^2. If there is no horizon in the frame, the value of the R^2 will be of the type *NaN*, and it will be used in future steps to determine if it is necessary to remove the ground from the picture.

5.5.2 Obstacle detection

The obstacle detection is performed by detecting the edges of them through the *edge* Matlab function, which results in a binary image with the edge pixels of the object candidates to be an obstacle with the value of 1 and the rest 0. The objects are filled and then, if there is ground in the image, determined by the R^2 as said, it is removed by a function that uses the output from the sky segmentation function. Therefore, the part of the image below the horizon line is not considered for the detection.

After that, a filter based on the area of the pixel groups (ones between zeros) is used to remove isolated pixels thus the probability of false detection is less, establishing a threshold for that.

With the filtered image, the different group of pixels found in the image will determine the number of obstacles. For each group, the mean of the coordinates they occupy in the image will determine the centroid of that obstacle and the length of the major axis will determine the diameter. This data is collected in an easily interpretable array of size $numberObstacles + 3 \cdot numberObstacles$, being the first element the number of obstacles, the next three the X coordinate, Y coordinate and radius of the obstacle, respectively, and so on until reaching the total number of obstacles.

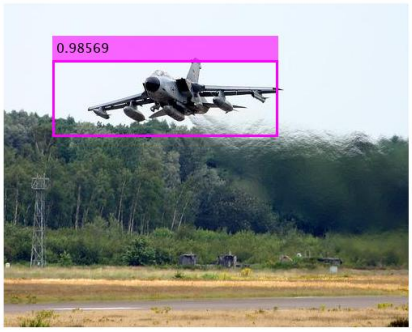
5.6 Results

In this section, both implementations are run over some of the images in the data base reserved for testing and the results are presented. Representatives cases have been selected in order to extract good conclusions about the advantages and limitations of each method. The obstacle detected is labeled in the case of the faster R-CNN with the confidence value in the detection. Computational time (obtained by running it several times and computing the mean) is annotated in the caption of each figure. In Figures 5.11, 5.12 and 5.13, first column shows the results for the faster R-CNN implementation while the second shows the results for the SVM + edge detection implementation. Again, the characteristics of the computer where the results are obtained from are AMD Ryzen 5 2600X Six-Core (3.60 GHz) CPU, 16 GB RAM and NVIDIA Quadpro P2000 GPU.

The images in Figure 5.11 shows some results in different generic scenarios.

But what happens if the obstacles are too close? Figure 5.12 shows the results in this case, where the implementation of SVM + edge detection algorithm is most likely to struggle. Data association will be needed to determine which detections correspond to each target.

On the contrary, if they are too far, shown in Figure 5.13, the faster R-CNN does not get the same amount of confidence when detecting and even it can not distinguish the properties of an aircraft in some cases and thus the detection fails.

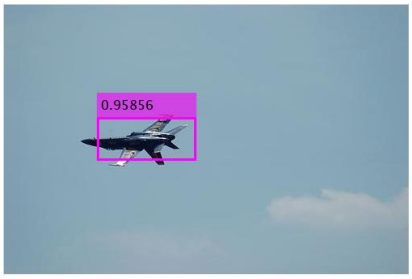


(a) Average time = 0,4680 s

(b) Average time = 7,0793 s

(c) Average time = 0,3856 s

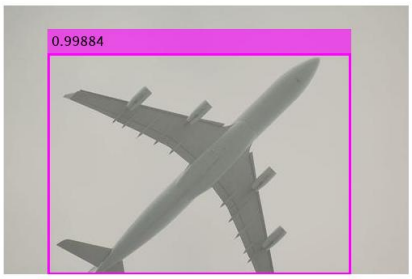
(d) Average time = 6,9824 s



(e) Average time = 0,3598 s

(f) Average time = 7,4521 s

Figure 5.11: Results and detection time for different images



(a) Average time = 0,3361 s

(b) Average time = 7,2339 s

(c) Average time = 0,3726 s

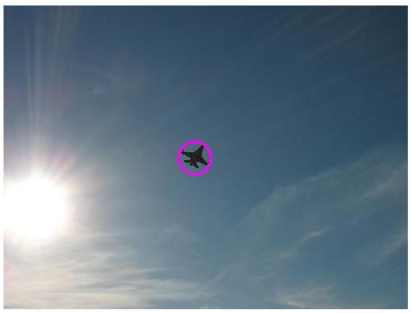
(d) Average time = 7,1355 s

(e) Average time = 0,3326 s

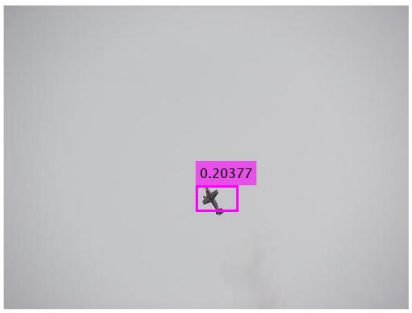
(f) Average time = 6,8890 s

Figure 5.12: Results and detection time for close obstacles

(a) Average time = 0,4112 s



(b) Average time = 6,8174 s



(c) Average time = 0,3824 s

(d) Average time = 7,2569 s

(e) Average time = 0,2123 s

(f) Average time = 7,4500 s

Figure 5.13: Results and detection time for far away obstacles

The implementation of the SVM + edge detection is not only meant to detect aircraft but also any obstacle above the horizon line. Thus, Figure 5.14 shows different results in those cases when something else is out there, such as a tree or a building. In order to detect those obstacles with the faster R-CNN algorithm, it would need to be trained for those classes (tree or building, for instance), therefore, it could be extended.

(a) Tree detections

(b) Building detections

Figure 5.14: Other obstacle detection with the SVM + edge implementation

Table 5.2 summarizes the main characteristics of each implementation.

	Faster R-CNN	SVM + edge detection
Average Time [s]	0,3623	7,1441
Range	Short - medium	Medium - long
Other obstacles	No (needs to be trained for more classes)	Yes (only above the horizon line)

Table 5.2: Comparison of both implementations

Therefore, the next conclusion can be reached: to use the implementation of the faster R-CNN as the main resource for aircraft detection, since it is faster and provides better results at short ranges, and to use the implementation of the SVM + edge detection to be run time to time and warn the system of distant threats. Nevertheless, other configurations can be implemented:

- A faster R-CNN trained for different types of obstacles will overcome the limitation of only detecting aircraft, but it will still perform detections in short-medium range if the time is not compromised.
- An edge detection algorithm (without the SVM capability to perform sky segmentation) if the camera is always pointing to the clear sky, that will overcome the time limitation, performing detections in a range of 0,1 to 0,2 seconds.

Chapter 6

Simulation environment

To test a functional SAA system, a physical robot needs to be assembled and programmed. In the Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence project developed at CfAR, the robot shall be a quadrotor UAV. In this chapter, Section 6.1 presents the chosen way to control the robot: Robot Operating System (ROS). However, real robots require logistics including lab space, recharging of batteries, operation licensing and have the risk of damaging both itself and people/property around it. Thus, it is a smart approach to simulate a robot and validate the algorithms by using simulators. Section 6.2 presents the simulator chosen to test the UAS: Gazebo. Finally, Section 6.3 shows how all of this can be used to test the algorithms implemented in Chapter 5.

This chapter aims to provide some insight into both the testing and simulation strategies that will be used to validate the SAA system.

6.1 ROS

ROS is an open source robotic software system that can be used without licensing fees. By being an open source software, there is access to the source code and one can modify it according to their needs, being possible to contribute to the software by adding new modules. One of the main purposes of ROS is to provide communication between the user, the computer's operating system and the external hardware (sensors, robots, etc). One of the benefits of ROS is the hardware abstraction and its ability to control a robot without the user having to know all of the details of the robot. For instance, to move a robot around, a ROS command can be issued, or custom scripts in *Python* or C^{++} can cause the robot to respond as commanded. The scripts can, in turn, call various control programs that cause the actual motion of the robot respecting its kinematics and dynamics limitations.

Essentially, ROS is a framework for writing robot software, having already a collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behavior

across a wide variety of robotic platforms [58]. The software is structured as a large number of small programs that rapidly pass messages between each other. Creating a robust robot software is a difficult task and ROS was built from the ground up to encourage collaborative robotics software development. This way, different organizations can make use of each other expertise, allowing faster development of complex robots.

6.1.1 ROS Graph

One of the original problems that motivated the development of ROS was a ”fetching robot” case. The solving of this problem led to several observations about many robotics software applications, which became some of the design goals of ROS [59]:

- The application task can be decomposed into many independent subsystems (e.g. navigation, computer vision, grasping, etc.);
- These subsystems can be used for other tasks (e.g. security patrol, cleaning, delivering mail, etc.);
- With proper hardware and geometry abstraction layers, the vast majority of application software can run on any robot.

The design goals can be illustrated by the fundamental rendering of a ROS system: its graph. A simple ROS graph is represented in Figure 6.1, where the ellipsis represents nodes and the arrows represent message streams between nodes; i.e. the edges.

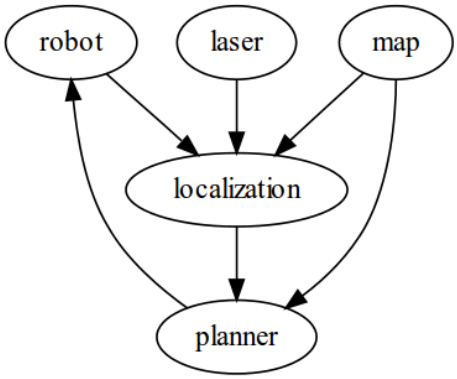


Figure 6.1: Example of a ROS graph

A ROS system is made up of a variety of programs, called *nodes*, running simultaneously and communicating with one another via *messages*. The ROS design idea is that each node is an independent module that interacts with other node using ROS communications capabilities. For that purpose, ROS makes use of `roscore` so nodes can find each other and transmit messages. Every node connects to `roscore` at startup to register details of the message streams it publishes and the streams it wishes to subscribe. Messages between nodes are transmitted peer-to-peer and `roscore` is only used by nodes to know where to find their peers.

6.1.2 ROS Packages, ROS Topics and `rosrun`

ROS is organized into *packages* which contain a combination of data, code and documentation. The ROS ecosystem includes thousands of publicly available packages in open repositories, encouraging its use between interested parties.

Although a ROS system consists of a number of independent *nodes* forming a *graph*, these nodes by themselves are typically not very useful. The nodes become relevant when they start communicating, exchanging information and data. The most common way to do that in ROS is through *topics*, which consist of a stream of messages with a defined type. In ROS all messages on the same topic must be of the same data type. A good practice is to name it in a way that describes the messages that are being transmitted.

Topics implement a *publish/subscribe* communication mechanism, which is one of the most common ways to exchange data in a distributed system. Before nodes start to transmit data over topics, they must first *advertise*, i.e. announce, both the topic name and the types of messages that are going to be sent. Then, they can start to *publish*, i.e. send, the actual data on the topic. Nodes that want to receive messages on a topic can *subscribe* to that topic by requesting `roscore`.

Since ROS has a sparse community, its software is organized in packages that are independently developed by community members. Chasing down long paths towards the packages in order to execute functions would become tiresome in large file systems since nodes can be deeply buried in large directory hierarchies. To automate this task, ROS provides a command-line utility, called `rosrun`, that will search a package for the requested program and pass it any parameters supplied.

6.2 Gazebo

As said in the introductory note, a lot of the trouble related to operate real robot can be avoided by using simulated robots. Although, at first glance, this seems to defeat the whole purpose of robotics, software robots are extraordinarily useful. In simulations, it can be model as much or as little of reality as desired. Sensors and actuators can be modeled as ideal devices or can incorporate various levels of distortion, errors and unexpected faults. Also, due to the nature of the final system developed at CfAR (path planning and control algorithms), it typically requires simulated robots to test the algorithms under realistic scenarios in a safe way.

Thanks to the isolation level provided by the messaging interfaces of ROS, the majority of the robot's software graph can be run identically whether it is controlling a real robot or a simulated one. The various nodes are launched and simply find one another and establish connections. Simulation input and output streams connect to the graph in the place of the device drivers of the real robot. Although some tuning is often required when transitioning to a physical robot, ideally the structure of the software will be the same, and often the simulation can be modified to reduce the number of parameter tweaks required

when transitioning between simulation and reality [58].

The chosen simulator for this project was Gazebo, which is a 3D physics simulator based on *rigid-body* dynamics, allowing for a good computational performance. Gazebo is capable of real-time simulation with relatively simple worlds and robots and, with some care, can produce physically plausible behavior.

ROS integrates closely with Gazebo through the `gazebo_ros` package, which provides a Gazebo plugin module that allows bidirectional communication between Gazebo and ROS. Simulated sensor and physics data can stream from Gazebo to ROS and actuator commands can stream from ROS back to Gazebo. In fact, by choosing consistent names and data types for the data streams, it is possible for Gazebo to exactly match the ROS API of a robot. When this is achieved, all of the robot software above the device-driver level can be run identically both on the real robot, and (after parameter tuning) in the simulator.

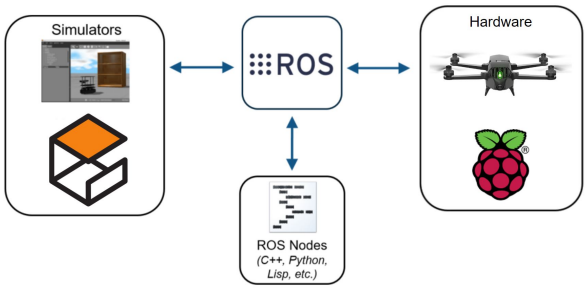
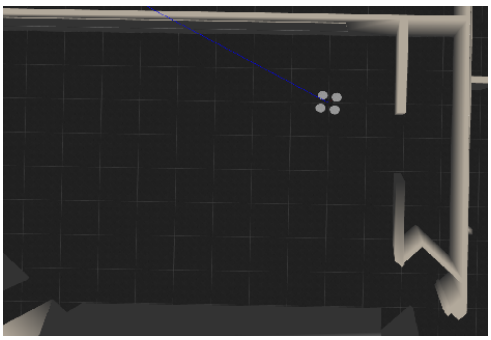


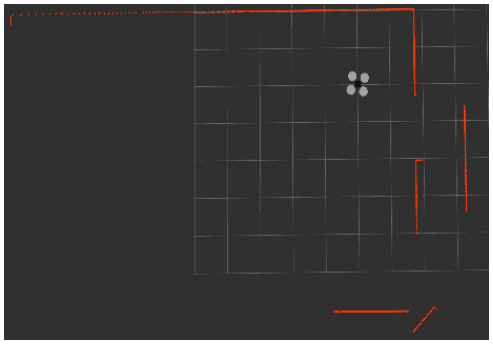
Figure 6.2: Prototyping

For the current application, Gazebo and the availability of ROS packages allow the integration of different sensors into a quadrotor model and an easy access to their outputs. Figure 6.3(a) shows a simple model of a quadrotor carrying an optical camera and a LIDAR, spawned inside an office. Also, it is compared what the operator perceives of the environment in Figure 6.3(b) (walls and free passages) with what the robot senses through the LIDAR in Figure 6.3(c) (an obstacle which reflected the light identified in red, or a path with no obstacle). Figure 6.3(d) shows the access to the hardware information, in this case, the image from a visual camera.

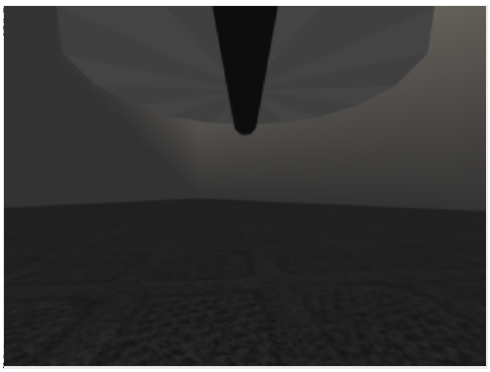
(a) Gazebo scenario with a default quadrotor



(b) Upper view of the environment



(c) LIDAR-mapped environment



(d) Image received from the visual camera

Figure 6.3: Quadrotor carrying an optical camera and a LIDAR and their information

6.3 Simulation

In order to test the capability of the UAS to acquire pictures of the environment and classify them regarding its obstacles, a simulation in Gazebo was conducted. In order to set the simulation, a few parameters need to be set:

- A world with obstacles.
- An UAV with a visual camera.
- A script to save and analyze the data collected.

6.3.1 World

A simple environment was set for simulation purposes, mainly due to the fact of the computational resources needed to run Gazebo. A simple scenario with blue sky and grass ground was firstly defined, to which some obstacles were added. Said obstacles encompassed a set of pine trees already modeled within the Gazebo Graphical User Interface (GUI). Regarding the airborne obstacles, an online model was used [60]. This model was selected based on its resemblance with a commercial aircraft while having smaller dimensions, making the simulation run smoother. Nevertheless, it could be improved choosing one more similar to commercial aircraft. Two of these UAVs were spawned and set to describe a lift-off-like maneuver in a loop. An example of the world is depicted in Figure 6.4.

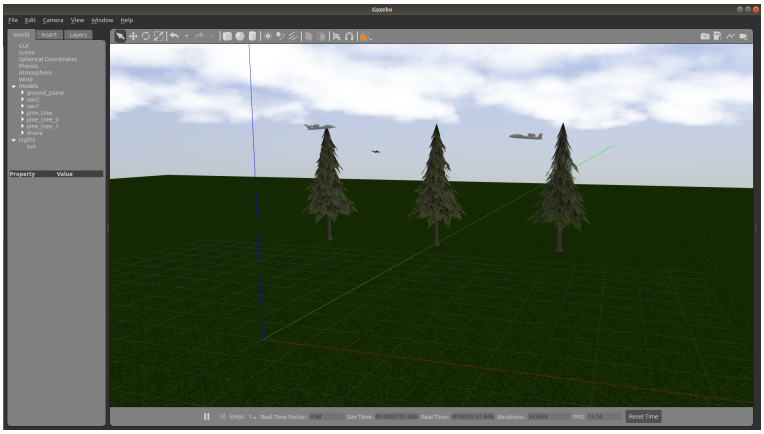


Figure 6.4: Simulation environment

6.3.2 UAV

To simulate the UAV, a simple CAD model was developed and converted to a file type that Gazebo can interpret as a solid body subjected to the laws of physics. In order to collect pictures of the environment, a camera model was also implemented with the characteristics listed in Table 6.1. The camera takes 30 pictures per second composed of squared pixels. Noise is modeled as a Gaussian process with zero mean and standard deviation of 0,007. The UAV used in the simulation is represented in Figure 6.5.

Parameter	Value
Update rate [fps]	30
Resolution [pixels x pixels]	500 x 500
Pixel format	R8G8B8
Field of View [°]	114,6
Range [m]	200

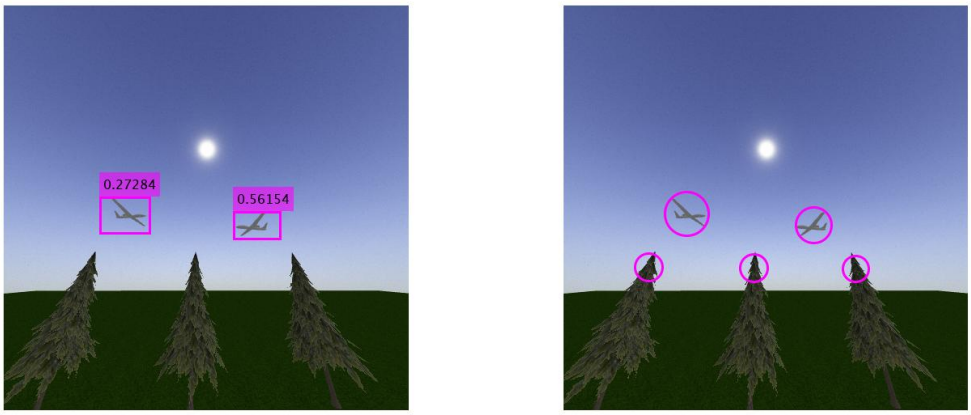
Table 6.1: Camera characteristics

Figure 6.5: Simulated UAV

6.3.3 Data collection

The images collected by the camera are published in the `image_raw` ROS topic. However, to analyze the pictures, it is necessary to convert them to a suitable format rather than a ROS message. To accomplish that, the open source computer vision library OpenCV [61] is used. The images taken were saved in a specific folder in order to be classified by the implemented algorithms. It was also possible to record multiple frames, save them and play them back as a video by using `rosbag`.

Some results of the implementations developed in Chapter 5 running over the images collected from this simulation environment are shown in Figure 6.6.



(a) Faster R-CNN on the simulation environment (b) SVM + edge detection on the simulation environment

Figure 6.6: Algorithms run over images collected from the simulation environment

It can be seen that although the intruder UAV model is not the most similar to a commercial aircraft, they are detected in both cases, as well as the tops of the trees overlooking the horizon in the case of the SVM and edge detection implementation, thus proving Gazebo's ability to be the platform where to test the developed algorithms.

Chapter 7

Conclusion

This document enfolds the research done for the Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence project. In Chapter 2, the most important concepts and theoretical aspects of SAA were mentioned and an architecture for the project was proposed. Chapter 3 and Chapter 4 explain the basis of the different non-cooperative sensors and data fusion algorithms respectively, pointing the aspects to take into account depending on the final application. The implementation of machine and deep learning to provide a detection capability with visual cameras is rigorously discussed in Chapter 5, including a comparison of both techniques. Finally, Chapter 6 discusses the most important aspects of the simulation environment that will be used to test all the algorithms within the project, including an example of an environment used to test the image processing algorithms developed in this thesis. All of which makes a good start to the project.

7.1 Achievements

The definition and direction of the project are now clear.

Among the different non-cooperative sensors, three are particularly suitable for the project: radars, LIDARs and visual EO cameras. Some guidance is given in when to use each one. Combinations between radar and camera for long range operations or LIDAR and camera for short range operations are desirable since they can complement each other's limitations. An interesting approach to the camera and radar combination was presented in Section 4.6, where the radar limitation in terms of resolution is complemented by the camera, while allowing a low computational cost since the block in charge of processing the information provided by the camera does not have to work on the entire image.

Chapter 4, dedicated to the data fusion, is a complete guide for the future researcher, addressing the steps to follow and the aspects to take into account for the application within the frame of this project. A good choice of algorithms could be k-means for data association, EKF for state estimation and fuzzy logic to make the decision.

Related to the experimental work carried out in Chapter 5, on one hand, it has been proved that using machine learning is a good technique to remove the ground clutter in the image, nevertheless the algorithm stills being computationally heavy. On the other hand, deep learning techniques show a good performance to detect aircraft plus it can be trained for other obstacle categories. A thorough study in which parameters to use to ensure a properly trained detector has been accomplished in order to obtain the best results. In terms of the results, the implementation of the faster R-CNN is a robust solution for short range operations while the implementation of the SVM and edge detection algorithm obtains better results in long range although it takes more time to compute, thus, a time to time scan with this implementation could be an option while using the faster R-CNN as a primary source of information.

Finally, a simulation environment is being prepared. ROS and Gazebo are a good frame for robot operations and simulations in the context of the project, thanks to the possibility of extending the algorithms tested there to a real robot and the easy integration of different sensing devices, such as a camera or a LIDAR.

7.2 Future Work

The most immediate activity recommended as future work regards finishing the experimental work by extending the Matlab algorithms to be used with ROS and Gazebo in real time. Using sockets or compiling the code in *Python* or C^{++} can serve to achieve this purpose.

Regarding the faster R-CNN implementation, it would be desirable to extend it to other classes of obstacles, such as birds, trees or buildings. For the SVM and edge detection implementation, it would be interesting trying to achieve a faster performance by tuning the parameters used to train the SVM.

The next step on the project would be to test other sensors such as a LIDAR or a radar to figure out how to process their information and, once done it, follow the advices in Chapter 4 to perform data fusion with the information given by the camera processing carried out in this thesis.

Eventually, the information of the final state of the obstacles obtained after the data fusion will be sent to the online path planning algorithm, where a trajectory to avoid the obstacles is generated and optimized. Integrating the navigation sensors' information within this framework would be desirable as well because the final goal of the Enhanced Guidance, Navigation and Control for Autonomous Air Systems based on Deep Learning and Artificial Intelligence project is to design an effective SAA system for real UAS and thus all this information would be sent to the flight management system that would send the flight plan to the autopilot. Therefore, work in this area could be also done.

Bibliography

[1] S. Waterman. Drones over u.s. get ok by congress. The Washington Times, https://www.washingtontimes.com/news/2012/feb/7/coming-to-a-sky-near-you/, February 2012.

[2] P. Angelov. *Sense and Avoid in UAS*. Wiley, 1^{st} edition, 2012. ISBN:978-0-470-97975-4.

[3] G. Fasano, D. Accardo, A. Moccia, and D. Moroney. Sense and avoid for unmanned aircraft systems. *IEEE Aerospace and Electronic Systems Magazine*, 31:82–110, 11 2016. doi: $10.1109/\mathrm{MAES}.2016.160116$.

[4] F. S. Team. Airspace, special use airspace and tfrs. Federal Aviation Administration, https://www.faasafety.gov/gslac/ALC/course_content.aspx?cID=42&sID=505.

[5] sUAS Aviation Rulemaking Committee. *Comprehensive set of recommendations for sUAS regulatory development*. Apr. 2009.

[6] S. Ramasamy, R. Sabatini, and A. Gardi. Avionics sensor fusion for small size unmanned aircraft sense-and-avoid. In *2014 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 271–276, May 2014. doi: $10.1109/\mathrm{MetroAeroSpace}.2014.6865933$.

[7] R. Arteaga. Automatic dependent surveillance broadcast ads-b sense-and-avoid system. NASA, https://ntrs.nasa.gov/search.jsp?R=20160007776, June 2013.

[8] C. T. Howell, T. M. Stock, P. J. Wehner, and H. A. Verstynen. Simulation and flight test capability for testing prototype sense and avoid system elements. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 8A1–1–8A1–16, Oct 2012. doi: $10.1109/\mathrm{DASC}.2012.6382430$.

[9] F. Castanedo. A review of data fusion techniques. *TheScientificWorldJournal*, 2013:704504, 10 2013. doi: $10.1155/2013/704504$.

[10] M. McLean. What polar pattern should i record my podcast with? The Podcast Host, https://www.thepodcasthost.com/equipment/microphone-polar-patterns/, July 2016.

[11] G. Fasano, D. Accardo, A. Moccia, G. Carbone, U. Ciniglio, F. Corraro, and S. Luongo. Multisensor based fully autonomous non-cooperative collision avoidance system for uavs. *Journal of Aerospace Computing, Information and Communication*, 5:338–360, 10 2008. doi: $10.2514/1.35145$.

[12] J. A. Jackson, J. D. Bošković, and D. Diel. Sensor fusion for sense and avoid for small uas without ads-b. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 784–793, June 2015. doi: 10.1109/ICUAS.2015.7152362.

[13] H. F. Durrant-Whyte. Sensor models and multisensor integration. *I. J. Robotic Res.*, 7:97–113, 12 1988. doi: 10.1177/027836498800700608.

[14] B. V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, Jan 1997. ISSN 0018-9219. doi: 10.1109/5.554206.

[15] R. C. Luo, Chih-Chen Yih, and Kuo Lan Su. Multisensor fusion and integration: approaches, applications, and future research directions. *IEEE Sensors Journal*, 2(2):107–119, April 2002. ISSN 1530-437X. doi: 10.1109/JSEN.2002.1000251.

[16] Huimin Chen, T. Kirubarajan, and Y. Bar-Shalom. Performance limits of track-to-track fusion versus centralized estimation: theory and application [sensor fusion]. *IEEE Transactions on Aerospace and Electronic Systems*, 39(2):386–400, April 2003. ISSN 0018-9251. doi: 10.1109/TAES.2003.1207252.

[17] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451 – 460, 1975. ISSN 0005-1098. doi: https://doi.org/10.1016/0005-1098(75)90021-7. URL http://www.sciencedirect.com/science/article/pii/0005109875900217.

[18] K. Chang, C. Chong, and Y. Bar-Shalom. Joint probabilistic data association in distributed sensor networks. In *1985 American Control Conference*, pages 817–822, June 1985. doi: 10.23919/ACC.1985.4788729.

[19] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979. ISSN 0018-9286. doi: 10.1109/TAC.1979.1102177.

[20] R. Streit and T. E. Luginbuhl. Maximum likelihood method for probabilistic multihypothesis tracking. *Proceedings of SPIE - The International Society for Optical Engineering*, 2235, 07 1994. doi: 10.1117/12.179066.

[21] I. J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, Feb 1996. ISSN 0162-8828. doi: 10.1109/34.481539.

[22] M. E. Liggins, Chee-Yee Chong, I. Kadar, M. G. Alford, V. Vannicola, and S. Thomopoulos. Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85(1):95–107, Jan 1997. ISSN 0018-9219. doi: 10.1109/JPROC.1997.554211.

[23] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering (ASME)*, 82D:35–45, 01 1960. doi: 10.1115/1.3662552.

[24] R. Luo and M. Kay. Data fusion and sensor integration: State-of-the-art 1990s. *Data fusion in robotics and machine intelligence*, pages 7–135, 1992.

[25] G. Bishop, G. Welch, et al. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8 (27599-23175):41, 2001.

[26] G. Fasano, D. Accardo, A. E. Tirri, A. Moccia, and E. D. Lellis. Radar/electro-optical data fusion for non-cooperative uas sense and avoid. *Aerospace Science and Technology*, 46:436 – 450, 2015. ISSN 1270-9638. doi: https://doi.org/10.1016/j.ast.2015.08.010. URL http://www.sciencedirect.com/science/article/pii/S1270963815002540.

[27] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, Oct 2000. doi: 10.1109/ASSPCC.2000.882463.

[28] R. Labbe. *Kalman and Bayesian Filters in Python*. 2015.

[29] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 138–149, New York, NY, USA, 2003. ACM. ISBN 1-58113-707-9. doi: 10.1145/958491.958508. URL http://doi.acm.org/10.1145/958491.958508.

[30] M. Manzo, T. Roosta, and S. Sastry. Time synchronization attacks in sensor networks. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '05, pages 107–116, New York, NY, USA, 2005. ACM. ISBN 1-59593-227-5. doi: 10.1145/1102219.1102238. URL http://doi.acm.org/10.1145/1102219.1102238.

[31] J. K. Uhlmann. Covariance consistency methods for fault-tolerant distributed data fusion. *Information Fusion*, 4(3):201 – 215, 2003. ISSN 1566-2535. doi: https://doi.org/10.1016/S1566-2535(03)00036-8. URL http://www.sciencedirect.com/science/article/pii/S1566253503000368.

[32] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, March 2002. ISSN 1053-587X. doi: 10.1109/78.984773.

[33] J. Martinez-del-Rincon, C. Orrite-Urunuela, and J. E. Herrero-Jaraba. An efficient particle filter for color-based tracking in complex scenes. In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 176–181, Sep. 2007. doi: 10.1109/AVSS.2007.4425306.

[34] Y. Bar-Shalom. Update with out-of-sequence measurements in tracking: exact solution. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):769–777, July 2002. ISSN 0018-9251. doi: 10.1109/TAES.2002.1039398.

[35] M. Orton and A. Marrs. A bayesian approach to multi-target tracking and data fusion with out-of-sequence measurements. In *IEE Target Tracking: Algorithms and Applications (Ref. No. 2001/174)*, volume Seminar, pages 15/1–15/5 vol.1, Oct 2001. doi: 10.1049/ic:20010241.

[36] M. L. Hernandez, A. D. Marrs, S. Maskell, and M. R. Orton. Tracking and fusion for wireless sensor networks. In *Proceedings of the Fifth International Conference on Information Fusion. FUSION*

2002. (IEEE Cat.No.02EX5997), volume 2, pages 1023–1029 vol.2, July 2002. doi: 10.1109/ICIF.2002.1020924.

[37] P. C. Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936.

[38] S. J. Julier, J. K. Uhlmann, and D. Nicholson. A method for dealing with assignment ambiguity. In *Proceedings of the 2004 American Control Conference*, volume 5, pages 4102–4107 vol.5, June 2004. doi: 10.23919/ACC.2004.1383951.

[39] C. Coue, T. Fraichard, P. Bessiere, and E. Mazer. Multi-sensor data fusion using bayesian programming: An automotive application. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 442–447 vol.2, June 2002. doi: 10.1109/IVS.2002.1187989.

[40] M. Liggins II, D. Hall, and J. Llinas. *Handbook of multisensor data fusion: theory and practice*. CRC press, 2017.

[41] G. M. Provan. The validity of dempster-shafer belief functions. *International Journal of Approximate Reasoning*, 6(3):389 – 399, 1992. ISSN 0888-613X. doi: https://doi.org/10.1016/0888-613X(92)90032-U. URL http://www.sciencedirect.com/science/article/pii/0888613X9290032U.

[42] A. P. Dempster. *A Generalization of Bayesian Inference*, pages 73–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-44792-4. doi: 10.1007/978-3-540-44792-4_4. URL https://doi.org/10.1007/978-3-540-44792-4_4.

[43] G. Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

[44] H. Wu and M. Siegel. Sensor fusion using dempster-shafer theory. 03 2003.

[45] Huadong Wu, M. Siegel, and S. Ablay. Sensor fusion using dempster-shafer theory ii: static weighting and kalman filter-like dynamic weighting. In *Proceedings of the 20th IEEE Instrumentation Technology Conference (Cat. No.03CH37412)*, volume 2, pages 907–912 vol.2, May 2003. doi: 10.1109/IMTC.2003.1207885.

[46] G. Zhao, X. Xiao, J. Yuan, and G. W. Ng. Fusion of 3d-lidar and camera data for scene parsing. *Journal of Visual Communication and Image Representation*, 25(1):165 – 183, 2014. ISSN 1047-3203. doi: https://doi.org/10.1016/j.jvcir.2013.06.008. URL http://www.sciencedirect.com/science/article/pii/S1047320313001211. Visual Understanding and Applications with RGB-D Cameras.

[47] H. Durrant-Whyte and T. C. Henderson. *Multisensor Data Fusion*, pages 585–610. 01 2008. doi: 10.1007/978-3-540-30301-5_26.

[48] L. Forlenza, G. Fasano, D. Accardo, and A. Moccia. Flight performance analysis of an image processing algorithm for integrated sense-and-avoid systems. *International Journal of Aerospace Engineering*, 2012, 06 2012. doi: 10.1155/2012/542165.

[49] M. Copeland. What's the difference between artificial intelligence, machine learning, and deep learning? Nvidia, https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/, July 2016.

[50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[51] S. Saha. A comprehensive guide to convolutional neural networks—the eli5 way. Towards Data Science, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, December 2018.

[52] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[53] R. Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[54] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6): 1137–1149, June 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2577031.

[55] R. Gholami and N. Fakhari. Chapter 27 - support vector machine: Principles, parameters, and applications. In P. Samui, S. Sekhar, and V. E. Balas, editors, *Handbook of Neural Computation*, pages 515 – 535. Academic Press, 2017. ISBN 978-0-12-811318-9. doi: https://doi.org/10.1016/B978-0-12-811318-9.00027-2. URL http://www.sciencedirect.com/science/article/pii/B9780128113189000272.

[56] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[57] C. Wolf. Position estimation based on a horizon. https://es.mathworks.com/matlabcentral/fileexchange/52955-position-estimation-based-on-a-horizon.

[58] M. Quigley, B. Gerkey, and W. D. Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015.

[59] C. Fairchild and T. L. Harman. *ROS robotics by example*. Packt Publishing Ltd, 2016.

[60] E. Gonzalez. Uav/uas concept. https://grabcad.com/library/uav-uas-concept-1, June 2019.

[61] Opencv. https://opencv.org, July 2019.

[62] L. Rodriguez Salazar, R. Sabatini, A. Gardi, and S. Ramasamy. A novel system for non-cooperative uav sense-and-avoid. 04 2013.

Appendix A

Sensor models

This appendix aims to show some of the radar, LIDAR and camera sensors considered during the research and presented as candidates for the project. The most important aspects of each one will be highlighted.

A.1 Radar models

The first one (A.1) is a radar from Fortem Technologies which main characteristic is that already provides the track to the user, thus the information is processed on the same radar and it is not needed a module apart. Also, it counts with a considerable range.

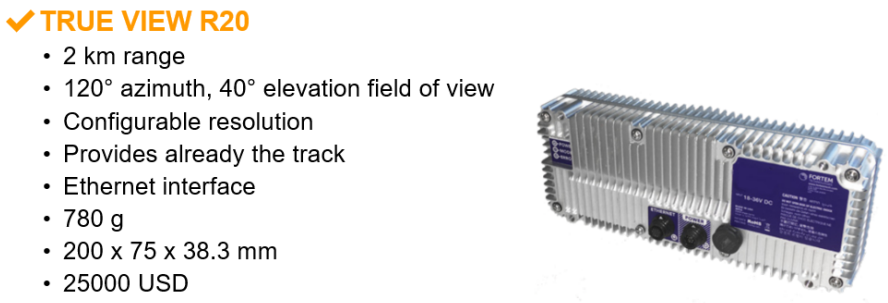
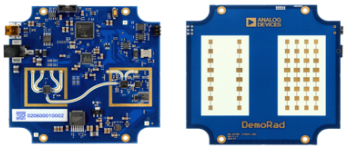


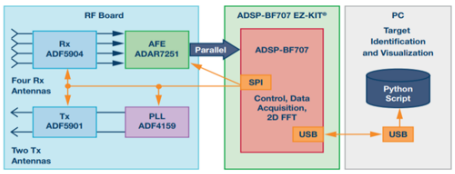
Figure A.1: True View Radar

The second one (A.2) is from Analog Devices and its presented as a kit. Nevertheless, the Demorad kit is not really intended as a turnkey solution, rather an evaluation platform or starting point for developing your own radar system, for prototyping radar for the application and for developing radar algorithms. The Demorad board is quite light to be mounted on the UAV but the board requires a supply and a USB connection to the processor for radar data transfer and data processing.

✔ DEMORAD

- 200m range
- 120° azimuth, 15° elevation field of view
- Digital beam forming
- 24 GHz
- Provides range, speed
- USB interface
- 60 g (only the substrate)
- 100 x 100 x 0.25 mm (only the substrate)
- 3000 USD





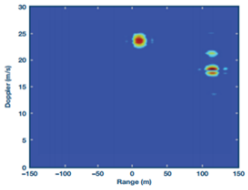


Figure A.2: Demorad Radar

The next radar (A.3) is already available at CfAR and it has been used in other projects.

✔ Chemring Altimeter

- Wide field of view
- Provides distance
- Serial interface

Figure A.3: Chemring Radar

The last one (A.4), from Aerotenna, stands out for its 360º FOV in azimuth, nevertheless, its range is too low which limits its applications.

✔ 360º Sense and Avoid

- 40 m range
- 360º azimuth field of view
- 24 GHz
- 243 g
- 115 x 115 x 67 mm

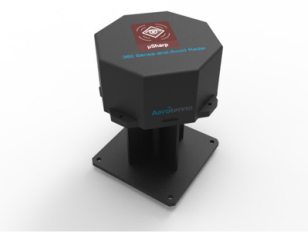


Figure A.4: 360º Sense and Avoid Radar

A.2 LIDAR models

Regarding the LIDAR models, CfAR has the M8 LIDAR (A.5), from Quanergy.

✔ M8 LIDAR

- >100 m range, < 3 cm accuracy
- 360º horizontal, 20º vertical field of view
- 0.03 to 0.13 angular resolution
- 905 nm wavelength
- 103 diameter x 87 mm height
- 900 g
- 420000 points per second (single return)

Figure A.5: M8 LIDAR

There are also upgraded models of this sensor which main difference is a longer range, up to 200 m. The M8 outputs the angle, distance, intensity and synchronized timestamps of each point in the cloud.

Another LIDAR candidate could be the Ultra Puck (A.6), from Velodyne, where each packet contains:

- Time of flight distance measurement
- Calibrated reflectivity measurement
- Rotation angles
- Synchronized time stamps

✔ Ultra Puck LIDAR

- 200 m range, ± 3 cm accuracy
- 360º horizontal, 40º vertical field of view
- 0.33º vertical resolution
- 0.1º to 0.4º horizontal resolution
- 903 nm wavelength
- 103 diameter x 86.9 height mm
- 925 g
- 600000 points per second (single return)

Figure A.6: Ultra Puck LIDAR

A.3 Camera models

The Flea3 line of USB3 Vision, GigE Vision and FireWire cameras, from FLIR, offers a variety of CMOS and CCD image sensors in a compact package. The Flea3 leverages a variety of Sony, ON Semi, and

e2v sensors ranging from 0,3 MP to 5,0 MP and from 8 FPS to 120 FPS. They have been used in some SAA system like the one described in [62]. Nevertheless, there have been some improvements and the latest sensors and most advanced feature sets can be found in the Blackfly S and Oryx families, also from FLIR. There are a lot of different models with different characteristics within each family.

On the one hand, Blackfly S cameras include both automatic and precise manual control over image capture and on-camera pre-processing and they are available in GigE, USB3, cased, and board-level versions. Their megapixels range from 0,4 to 20 and the frames per second from 18 to 522, for instance. Each model also has different pixel size and resolution, and it can be found in mono or color, although for the current computer vision application it would be necessary a color camera.

Figure A.7: Blackfly S USB3 camera

On the other hand, Oryx cameras are only available with GigE (Gigabit Ethernet) interface. They range from 27 to 162 FPS and 5 to 31 MP. As in the previous family, each model also has a different pixel size and resolution, and it can be found in mono or color. Their main advantages are the IEEE1588 Clock Synchronization for precise timing across multiple devices and color correction matrix, gamma, saturation and sharpness that reduce host-side processing requirements.

Figure A.8: Oryx 10GigE camera