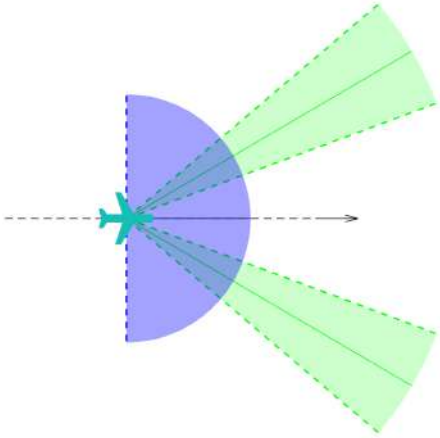
TÉCNICO LISBOA



Optimal Multi-Sensor Collision Avoidance System for Small Fixed-Wing UAV

Marta Ferro Barreto Candeias Portugal

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor: Prof. André Calado Marta

Examination Committee

Chairperson: Prof. Paulo Sérgio De Brito André
Supervisor: Prof. André Calado Marta
Member of the Committee: Prof. Pedro Da Graça Tavares Álvares Serrão

November 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Firstly, I want to express my gratitude towards Prof. André Marta for his incessant guidance and close supervision in every step of this thesis. It has truly been a privilege to work with someone who personifies IST's spirit and excellence, without ever overlooking its impact on students' personal growth.

I would also like to thank Andrew Brahim for his expert advice on PX4 sensor integration and Afonso Vale for his continuous help and problem solving along most stages of this work.

Finally, this master thesis, as the culmination of the last five years, would not have been possible without my friends' and family's support and the intercession of Our Lady throughout all my academic journey. A warm thank you to mom, dad, Maria, and Kiko.

Resumo

Este trabalho apresenta uma solução para o aumento da segurança de pequenos UAVs de asa fixa no que diz respeito à deteção de obstáculos durante o voo. O principal objetivo é implementar uma configuração do sistema multi-sensor ótima. Como tal, foram estudados trabalhos anteriores relativos à integração de sensores neste tipo de sistemas. Em seguida, foram modelados determinados sensores para simulações de deteção e prevenção de colisão, utilizando o método dos campos potenciais. Foi realizado um estudo de otimização através de um algoritmo genético, de maneira a encontrar os conjuntos de sensores e respectiva orientação que resultam num melhor desempenho no que toca a prevenção de colisões. Para o efeito, foi gerado aleatoriamente um conjunto de cenários de colisão com obstáculos fixos e móveis. Este estudo resultou em configurações de deteção relativamente simples que proporcionaram uma elevada taxa de sucesso na prevenção de colisões. O sensor ultrassónico revelou-se inadequado devido ao seu curto alcance, enquanto o sensor laser beneficia de um longo alcance, mas tem um campo de visão muito limitado. Em contrapartida, tanto o LIDAR como o RADAR são os mais promissores, pois apresentam um alcance significativo e um campo de visão alargado. As melhores configurações multi-sensor contêm um LIDAR ou RADAR frontal, complementado por um par de sensores laser laterais a 10 ou 63 graus, respetivamente. A montagem do sistema final, incluindo sensores e o controlador de voo PixHawk, foi então projetada e executada. O software apropriado (PX4, QGroundControl) foi também desenvolvido e adaptado ao presente trabalho. Para validar o sistema proposto, todos os sensores foram primeiro testados individualmente. Os testes comprovaram a exatidão das especificações dos sensores e das simulações anteriores. Seguiram-se testes num rover, utilizando um algoritmo simples de evasão de obstáculos, que por sua vez apresentou resultados satisfatórios.

Palavras-chave: Segurança, deteção de obstáculos, evasão de colisão, fusão de sensores, sensor laser, LIDAR, RADAR, otimização.

Abstract

This work provides a solution for the safety enhancement of small fixed-wing UAVs regarding obstacle detection during flight. The main goal is to implement an optimal multi-sensor system configuration. Therefore, preceding works regarding the integration of available sensors in such systems were studied. As a result, select sensors were modeled for collision detection and avoidance simulations using the potential fields method. An optimization study using a genetic algorithm was conducted to find the sets of sensors and respective orientation that result in the best collision avoidance performance. To do so, a set of collision scenarios with both stationary and moving obstacles were randomly generated. This study resulted in relatively simple detection configurations that provided high collision avoidance success rate. The ultrasonic sensor revealed to be inappropriate given its short range, while the laser rangefinder benefited from long range but had very limited field-of-view. In contrast, both the LIDAR and the RADAR are the most promising, as they exhibit a significant range and a broad field-of-view. The best multi-sensor configurations were either a front-facing LIDAR or RADAR, complimented by a pair of laser rangefinders pointing sideways at 10 or 63 degrees, respectively. The assembly of the final system, including sensors and a PixHawk flight controller, was then designed and executed. The appropriate software (PX4, QGroundControl) was also built and adapted to the current work. To validate the proposed system, all sensors were first individually tested. The bench tests attested the accuracy of the sensor specifications and previous simulations. Ground tests on a rover using a simple obstacle avoidance algorithm displayed satisfactory results.

Keywords: Safety, obstacle detection, collision avoidance, sensor fusion, laser rangefinder, LIDAR, RADAR.

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	True azimuth.
α_{PF}	Weighting term.
β	Sensor orientation.
θ	Angle between desired motion and obstacle.
λ	Bias compensation factor.
σ	Standard deviation.
ϕ	UAV banking angle // Objective function weighting term.
$\dot{\Omega}$	UAV turning rate.

Roman symbols

$\mathbf{c}$	Centre point of the sphere.
D	Filtered diameter (used measurement in filter).
d	Distance from the origin of the line.
$\mathbf{d}_0$	Vector pointing from object centre to the UAV.
d_{min}	Minimum distance to an obstacle.
$\mathbf{f}_{at}$	Attractive force.
$f(\beta)$	Fitness function.
G	LIDAR gain.
$\hat{\mathbf{k}}$	z axis unit vector.
$\mathbf{m}$	Direction of motion.
n	Number of filter cycles.
P	Point on the UAV path.

P_{close} Closest point on the path.

P_{next} Next point on the path.

p Fraction that represents the desired accuracy of filter dimensions.

r Radius // range.

$\mathbf{R_a}$ Action radius.

$\mathbf{R_c}$ Collision radius.

$\mathbf{R_d}$ Detection radius.

$\mathbf{R_s}$ Safety radius.

$\mathbf{s}$ Swirling unit vector.

S_{max} Maximum potential in repulsive fields.

t Time instant.

$\hat{\mathbf{u}}$ Unit vector that defines the line direction in 3D.

$\mathbf{x}$ Generic point.

x, y, z Cartesian position.

Subscripts

∞ Free-stream condition.

α Azimuth.

$cut-off$ Cut-off.

dir Directional component.

i, j, k Computational indexes.

k Current discrete measurement.

$k-1$ Previous discrete measurement.

m Measurement.

ref Reference condition.

sol Solution.

u Unbiased.

x, y, z Cartesian components.

Glossary

ADS-B	Automatic Dependent Surveillance-Broadcast
API	Application Programming Interface
BVLOS	Beyond Visual Line of Sight
CFR	Code of Federal Regulations
CMFK	Converted Measurement Kalman Filter
CPA	Closest Point of Approach
CR	Close Range
CW	Continuous Wave
EASA	EU Aviation Safety Agency
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
EU	European Union
FAA	Federal Aviation Administration
FMCW	Frequency Modulated Continuous Wave
FMU	Flight Management Unit
FOV	Field of View
GA	Genetic Algorithm
GPS	Global Positioning System
GPSAR	Ground Penetrating Synthetic Aperture RADAR
HALE	High Altitude Long Endurance
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
ISEP	Instituto Superior de Engenharia do Porto
ISR	Intelligence, Surveillance and Reconnaissance
KRISO	Korea Research Institute of Ships and Ocean Engineering
LADP	Low Altitude Deep Penetration
LALE	Low Altitude Long Endurance
LIDAR	Light Detection and Ranging

MALE	Medium Altitude Long Endurance
MR	Medium Range
MRE	Medium Range Endurance
MSRP	Manufacturer's Suggested Retail Price
MTOW	Maximum Takeoff Weight
NASA	National Aeronautics and Space Administration
PWM	Pulse-Width Modulation
PPM	Pulse-Position Modulation
RADAR	Radio Detection and Ranging
RCP	Robot Center Point
RCS	Radar Cross Section
RGB	Red Green Blue
RGB-D	Red Green Blue- Depth
RTK GNSS	Real Time Kinematic Global Navigation Satellite System
RTOS	Real-Time Operating System
S&A	Sense and Avoid
SCL	Serial Clock
SDA	Serial Data
SR	Short Range
UART	Universal Asynchronous Receiver-Transmitter
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UCAV	Unmanned Combat Aerial Vehical
UGV	Unmanned Ground Vehicle
UK	United Kingdom
uORB	Micro Object Request Broker
US	United States
USV	Unmanned Surface Vehicle
UTM	UAV Traffic Management
VFH	vector Field Histogram
VTOL	Vertical Takeoff and Landing
WP	Way-Point

Chapter 1

Introduction

This chapter presents an insight into the Unmanned Aerial Vehicle (UAV) industry by briefly overviewing the current market, discussing the background of the study and relevance of research in UAV safety systems. Finally, tangible objectives and deliverables are established, as well as the structure of the final work.

1.1 UAV Market Overview

UAVs have received considerable attention in a myriad of operations due to their enhanced stability and endurance. Despite being initially developed for military purposes; recently, there has been a notable upsurge in the civilian market for UAVs [1]. Accordingly, Fig.1.1 illustrates the growing revenue of the commercial UAV market in the USA in different applications. These applications include:

- **Precision agriculture**: Real-time imagery and maps are obtained during the decision-making process, enabling the use of remote sensing imagery to map changes in the field [2];

- **Energy sector monitoring**: Detection and prevention of faults in overhead electric power lines [3];

- **Security and law enforcement**: Detection, tracking and recognion of illegal activities, unwanted infiltrations, and unauthorized trespassers and prevention from unlawful cross-borders activities [4];

- **Media and entertainment**: Capturing otherwise inaccessible aerial photography and video footage due to UAV reduced size and high maneuverability.

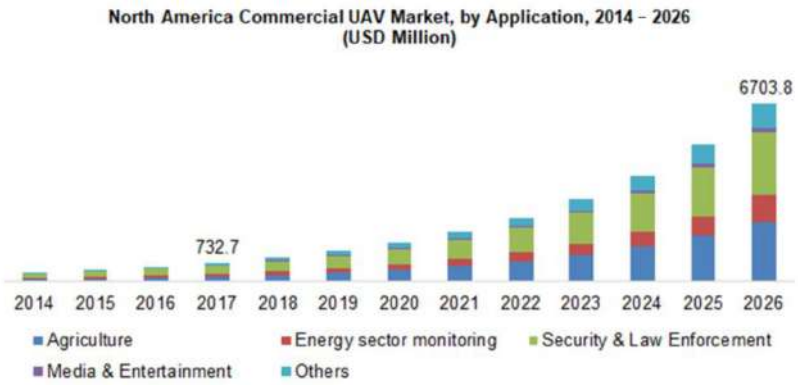


Figure 1.1: Commercial UAV market of North America [5].

In addition to its applications, a large number of metrics have been used to classify UAVs. These include maximum takeoff weight (MTOW), size, operating conditions, capabilities, or any combination of these and other characteristics. Table 1.1 is an illustration of the multiple criteria of differentiation possible. Although MTOW provides a good basis for classifying aircraft according to their risk to people and property following a ground impact, altitude-based UAV classes are also of interest since they will dictate collision avoidance requirements to some extent.

Table 1.1: Classification of UAVs [6].

Category	MTOW (kg)	Range (km)	Flight alt. (m)	Endurance (h)
Micro	<5	<10	250	1
Mini	<20/25/30/150	<10	150/250/300	<2
Close Range (CR)	25-150	10-30	3,000	2-4
Short Range (SR)	50-250	30-70	3,000	3-6
Medium Range (MR)	150-500	70-200	5,000	6-10
MR Endurance (MRE)	500-1,500	>500	8,000	10-18
Low Altitude Deep Penetration (LADP)	250-2,500	>250	50-9,000	0.5-1
Low Altitude Long Endurance (LALE)	15-25	>500	3,000	>24
Medim Altitude Long Endurance (MALE)	1,000-1,500	>500	3,000	24-48
High Altitude Long Endurance (HALE)	2,500-5,000	>2,000	20,000	24-48
Stratospheric (Strato)	>2,500	>2,000	>20,000	>48
Unmanned combat AV (UCAV)	>1,000	1,500	12,000	2

A simpler classification, only taking into account the level of human involvement is:

- **Remotely piloted**: A certified pilot remotely controls the UAV;
- **Remotely operated (semiautonomous)**: The UAV is given high-level flying commands (e.g., waypoints), but its performance is monitored by a trained operator, responsible for all decisions;
- **Fully autonomous**: The UAV is given general tasks and is capable of determining how to accomplish them, even at the face of unforeseen events and after the occurrence of faults.

Lastly, UAVs can be categorized in terms of wing configuration, having either fixed wing or rotary wing. The configuration must be chosen according to the operational mission at hand, as can be seen in Tab. 1.2.

Table 1.2: Key features of different categories of UAVs [1].

Types of UAVs	**Key Features**
Fixed-Wing	High speed, long endurance
Fixed-Wing Hybrid	Long endurance, VTOL
Single Rotor	Long endurance, hovering, VTOL
Multirotor	Short endurance, hovering, VTOL

For instance, long endurance is a key feature of small fixed-wing UAVs, as seen in Tab. 1.2. An example of a representative UAV is the Tekever AR4 (Fig. 1.2), an autonomous, fixed-wing mini UAV designed and manufactured in Portugal. This aircraft has an MTOW of 4 kg, including a payload capacity of 1 kg, an endurance of 2 hours, a maximum speed of 15 m/s and is hand launched for take-off [7].

Figure 1.2: Tekever AR4 [7].

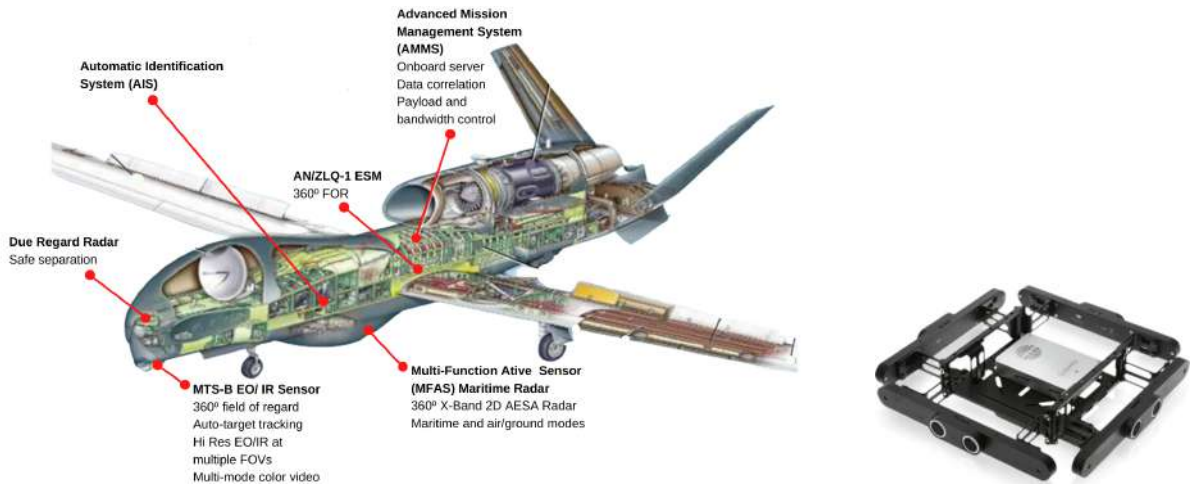
1.2 UAV Safety Systems

The forementioned rapid growth of the civilian UAV market places a great deal of responsability in UAV safety systems. These systems, like the safety of most aircrafts, strongly rely on sensors. Acting as a translator of the surroundings of the vehicle, sensors usually provide information related to position, velocity, acceleration and obstacle positions. Improvements in this area are critical to enhance safety and allow for the future integration of unmanned aircraft in the airspace.

Some applications of UAVs present high collision risk. Due to their ability to work in a collaborative and cooperative manner, swarms of drones are typically used for military and surveillance purposes, tracking and localizing objects. One of the most significant challenges regarding the navigation of a swarm of agents is collision avoidance. Collision avoidance systems are responsible for guiding an autonomous agent in order to safely and reliably avoid potential collisions with other agents in the swarm as well as with other objects in the environment. In addition to anti-aircraft weapons, a swarm of UAVs is vulnerable to power and communications link losses, posing as a collision threat to itself. The capacity to locally sense and avoid items in the environment becomes more crucial in order for agents to be fully autonomous, since not depending on a central server makes the system more robust. When numerous agents work together to complete a specified goal, such as navigation to the desired destination, this

local collision avoidance capability becomes even more vital [8].

Military and surveillance operations, flown either in swarms or individually, are also one of the biggest catalysts for unmanned aircraft system's (UAS) development. Northrop Grumman is currently working on the *MQ-4C Triton* UAS Program, which perfectly ilustrates the previous statement. Built for the US (United States) Navy, *Triton* will support a wide range of missions including maritime intelligence, surveillance and reconnaissance (ISR) patrol, signals intelligence, search and rescue and communications relay. Furthermore, it will be equipped with a unique and robust mission sensor suite that provides 360-degree coverage on all sensors (see Fig. 1.3(a)), providing unprecedented maritime domain awareness for the U.S. Navy. The aircraft can fly over 24 hours at a time, at altitudes higher than 10 miles, with an operational range of 8,200 nautical miles, providing real-time ISR over vast ocean and coastal regions [9].



(a) *MQ-4C Triton* by Northrop Grumman [9].

(b) *Guidance* by DJI [10].

Figure 1.3: Example of UAV sensing systems.

On the opposite side of the spectrum, as an elementary example, DJI developed a sensing system called *Guidance*, ensuring the collision avoidance of mini UAVs (paired with a flight controller). To do so, the system is equipped with a processing core, integrated visual cameras, ultrasonic sensors and computer vision algorithms. Tracking the area from all sights, alongside with data fusion techniques, *Guidance* composes a Sense and Avoid (S&A) system, as seen in Fig. 1.3(b) [10]. However, this solution is only available for multicopters, which typically fly at low speeds and have hovering capability.

Rather than focusing on sensing hardware, on a similar scale to that of the present work, an Automatic Dependent Surveillance–Broadcast (ADS-B) information-based collision avoidance methodology was developed in [11], involving two main steps. First, a UAV conflict-sensing scheme is developed, which utilizes ADS-B information flow path and analyzes the message format information. Second, an unscented Kalman filter is used to predict UAV trajectories based on the acquired ADS-B information. The predicted information is then used to determine potential conflict scenarios and different deconfliction strategies are selected accordingly. These strategies include speed regulation, direction regulation, and compound deconfliction. Although its primary objective was to improve the conflict resolution ca-

pability of UAV flights, this research also provides a valuable contribution to the field of UAV collision avoidance, serving as a theoretical foundation for further advancements in this area.

1.3 Motivation

As the UAV industry grows, airspace systems and regulations need to adapt. Thus, the Federal Aviation Administration (FAA) is carrying out a multibillion-dollar infrastructure program to modernize the US National Airspace System (NAS) called *NextGen* [12]. Civilian drones are subject to unique airworthiness categories under FAA regulation, in contrast to commercial passenger and cargo aircraft. Because of this, National Aeronautics and Space Administration (NASA) and the FAA have started working together to develop the UAV traffic management (UTM) platform, a project for an air traffic control system that simultaneously handles low-flying drones and manned aircraft.

Drones must exhibit a practical resolution for a S&A feature as part of the NextGen strategy for integrating UAVs into the national airspace. In fact, UAVs must deploy an automated S&A intelligent system that provides safety levels comparable to or even superior to those of manned aircraft, according to Title 14 Code of Federal Regulations (CFR) [13]. In a recent report [14] discussing the challenges and opportunities of UAS beyond visual line of sight (BVLOS) operations, the FAA once again highlighted the importance of establishing a new regulatory framework to capitalize on UAS, enhance safety, and promote sustainable transportation solutions.

As for the United Kingdom (UK), research also suggests that there is a need to establish equitable regulatory and technology environments relating to shared airspace for both drone and crewed aircraft operations. A study endorsed by the Journal of Air Transport Management [15] introduces the 'Class Lima' concept, which limits UAV operations to certain designated airspace zones but allows crewed aircraft to enter if they are carrying appropriate de-confliction equipment. From the perspective of further research, the need to investigate how the interaction between drones and general aviation aircraft might be achieved in the real-world is highlighted. In general, large-scale steps are currently being taken to enable the safe establishment of UAVs in the various types of airspaces, both in the UK and the US [16].

Several ongoing initiatives and new developments in the domain of European Union (EU) regulatory frameworks at various levels are reflected. In December 2020, the EU Aviation Safety Agency's (EASA) drone regulations for hobbyist and commercial drone operations became effective [17]. According to EASA, these regulations established requirements for three categories of drone operations (see Fig. 1.4): (1) Open (or "low-risk" operations) that do not require authorization but are subject to operational limitations; (2) Specific (or "medium-risk" operations) that require authorization from the national aviation authority on the basis of a risk assessment; and (3) Certified (or "high-risk" operations) that require certification and licensing.

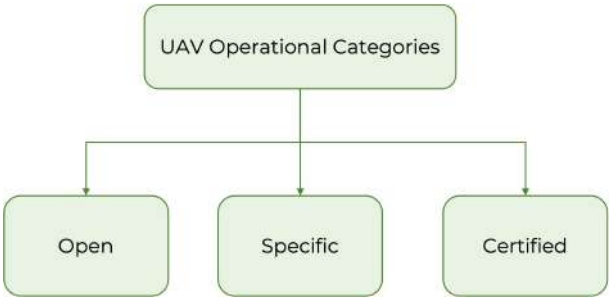


Figure 1.4: Categorization of UAS operations under EU regulation.

As of January 2023, EU rules establishing a dedicated airspace for drones known as the U-space became applicable [18]. The U-space creates conditions for more complex and longer-distance operations, particularly in low-level and densely operated airspace, and when out of sight of the remote pilot. As a new reality evolves towards comprehensive airspace integration, it is imperative for UAVs to adapt and correspond to safety standards, which in turn calls for research in innovation in this particular field.

Fittingly, this work addresses the safety enhancement of small fixed-wing UAVs (MTOW $<$ 25 kg, range $<$ 10 km, endurance $<$ 2h and flight altitude $<$ 120 m), particularly with regard to the detection of obstacles during flight and the automatically triggered collision avoidance maneuver. It is part of a comprehensive obstacle detection and collision avoidance system, representing a two-stage "sense" and "avoid" problem, being this work more focused on the former. The Sense and Detect stage is responsible for the acquisition of the necessary information that enables to detect, based on estimation techniques, collision threatening situations with either fixed or moving objects. The Collision Avoidance stage is responsible for replanning the flight path so that the UAV avoids the previously identified threats, taking into account the UAV dynamic and performance capability, as part of an optimal control problem.

1.4 Objectives and Deliverables

Two other master thesis preceed this work. In [19], different detection systems were simulated with a myriad of sensor types and configurations. Through the Potential Fields method and resorting to an optimization algorithm, the author reached a possible configuration of the UAV detection system. Subsequently, [20] followed up on that work, testing hardware and implementing an effective S&A System on a simple rover.

In this study, the main goal is to leverage the insights from the two aforementioned studies in order to implement an adapted and optimized version of said systems onto a rover, as an intermediate step towards generating a robust system to be employed in a small fixed-wing UAV. Taking advantage of pre-developed work done on the avoidance phase allows for the focus of this thesis to be almost entirely on the detection phase. Table 1.3 illustrates the proposed progress, compared to what has already been done.

Table 1.3: Precedent work comparison.

	Alturas [19]	**Serrano** [20]	**Current work**
Types of sensors modeled	Laser Rangefinder RADAR	Laser Rangefinder Ultrasonic Sensor	Laser Rangefinder Ultrasonic Sensor RADAR LIDAR
# of sensors to fuse data from	1	1	Multi
Design Configuration	Parametric	Parametric	Optimal
Implementation	-	Rover	Rover

The work can be divided into four main parts. After a comprehensive study of the sensors in question and respective models, different design solutions need to be examined in existing virtual environments. Once an optimal configuration is reached, the most promising solutions ought to be implemented in hardware and ground tested under controlled conditions. The end result would be a validated optimal sensing solution for a rover, including hardware and software components.

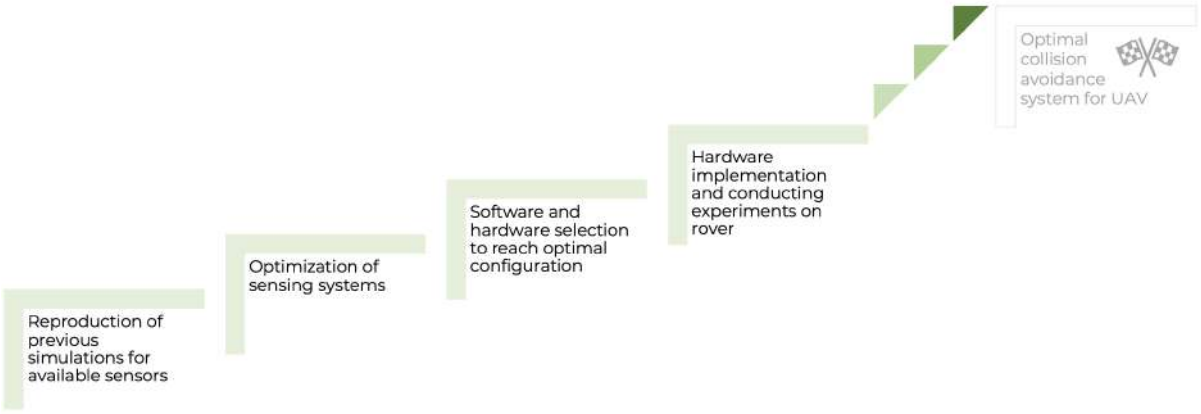


Figure 1.5: Process chart of present work.

The most important phases required to present a final product are visually represented in Fig. 1.5 and the objectives for this specific work are highlighted in green. Once the current goals are reached, transitioning from rover implementation to UAVs is a logical progression, which entails the adaptation and optimization of the developed systems for UAV platforms.

Chapter 2

Obstacle Sensing and Avoidance

This chapter provides an overview of the typical architecture of S&A systems, briefly explaining each module, and a review of the current state-of-the-art. Obstacle detection resorts to sensors, which can be divided into two categories: cooperative and non-cooperative. Since this work will focus on non-cooperative obstacle sensing, a deeper study on several sensors of this kind will be made, followed by a qualitative comparative analysis. Lastly, theoretical approaches for the collision algorithm are exposed and discussed.

2.1 Architecture of S&A Systems

A Sense and Avoid system is an automonous system capable of controlling the UAV in order for it to detect and avoid obstacles in a timely manner. This process can be divided into several modules represented in the functional block diagram in Fig. 2.1.

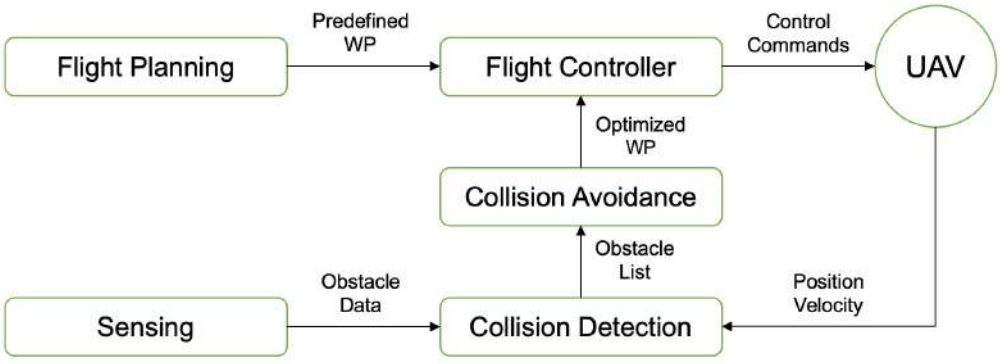


Figure 2.1: Block diagram of S&A systems' architecture. (adapted from [19].)

The Flight Planning module calculates the best route before takeoff using a set of specified way-points (WP). Cost functions that account path lengths, flight altitude, danger zones, energy consumption, threats and flight time are developed to determine the optimal route [21].

The list of waypoints to be followed are loaded into the Flight Controller once this stage is complete, providing control orders to the UAV actuators so that the UAV may execute the proper maneuver to move on to the next WP.

The Sensing module is in charge of continuously scanning the area around the UAV for obstacles and providing the obstacle data to the Collision Detection module on a regular basis so that it can be kept in a list. Given the position of the UAV, the Collision Detection module may calculate the distance to an obstacle and use that information to rank the list's components from nearest to farthest, with the closest obstacles posing the most risk and taking into account obstacles' velocity. The Collision Avoidance module will be activated if one of the obstacles from the list endangers the UAV's navigation, identifying possible solutions to avoid the threat. When developing this module, a number of factors must be taken into account, including the UAV's maneuverability, safety, cost of the path and energy to be consumed. These and other factors are incorporated either in the form of constraints that a path must adhere to or cost functions that must be minimized. As soon as a solution is identified, the waypoint list is updated to feature the new optimized waypoint and passed on to the flight controller module to allow the tracking of the new path [22].

All of these modules are necessary to the proper functioning of an S&A system. Nevertheless, the Sensing stage is a vital task and therefore constitutes the focus of this work. Currently, there are two types of approach to this problem: cooperative and non cooperative models. While the former implies the existence of a similar system on the intruder, the latter is independent.

This work aims at developing and implementing a S&A system for a small fixed-wing UAV, thus, [19] and [20] proposed the use of non-cooperative sensors to achieve this objective. The authors discarded cooperative sensing, as these models require other aircraft to be equipped with the same technology, disregarding static obstacles and unequipped aircraft. Furthermore, not only is non-cooperative detection lighter, but it is also not as costly.

2.2 State-of-the-art

S&A systems can be applied to different autonomous vehicles. This section is dedicated to a comprehensive exploration of the latest advancements, technologies, and pertinent research within this ever-evolving domain. By investigating the state-of-the-art in S&A systems, their versatile and adaptive nature is underscored, highlighting their role in addressing the safety and collision avoidance requirements across an array of autonomous vehicle platforms.

2.2.1 Unmanned Surface Vehicles (USV)

To ensure safety and reliability and to perform complex tasks autonomously, Unmanned Surface Vehicles (USVs) are also required to possess accurate perception of the environment and effective collision avoidance capabilities. In [23], a Radio Detection and Ranging (RADAR) based collision avoidance system was developed. The workflow of the hardware system is shown in Fig. 2.2. The marine

RADAR, which is mounted on top of the USV, collects environmental information within the RADAR range. The RADAR receiver converts the echo signals into digital images, which are then displayed on a screen, while video of RADAR images is output to the image acquisition card. The image acquisition card captures the video signal and sends the digital images to the RADAR target detection computer. The RADAR target detection computer performs image processing and analysis using the corresponding algorithms, and the processed environmental information is sent to the motion control and planning computer through network communication. With this information, the motion control and planning computer adjusts the real-time positioning of the USV through rational planning and control algorithms.

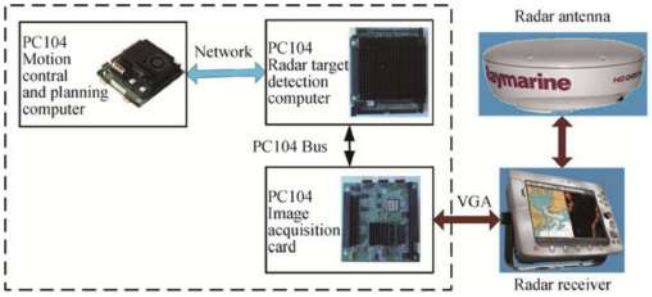


Figure 2.2: Hardware workflow of a RADAR-based collision avoidance system for a USV [23].

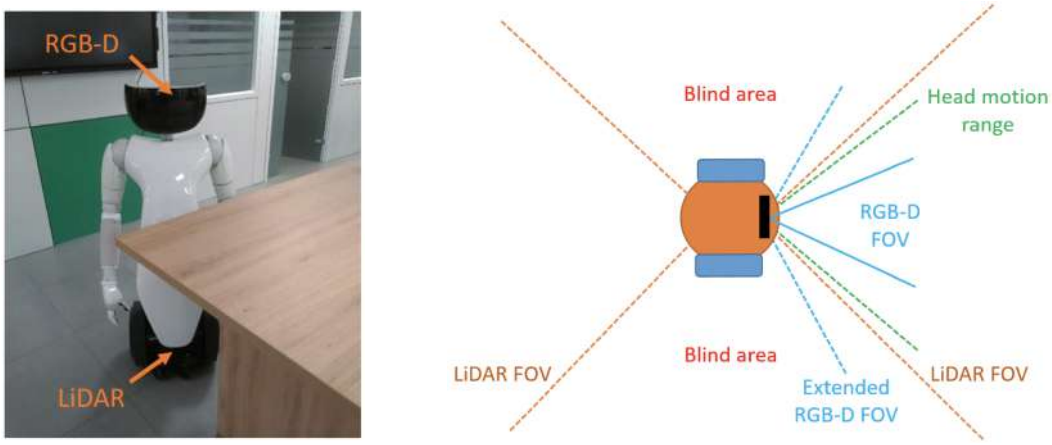
RADAR imagery techniques are typically exploited for USVs, as seen in Instituto Superior de Engenharia do Porto's (ISEP) *ROAZ II* [24]; and in Korea Research Institute of Ships and Ocean Engineering's (KRISO) *ARAGON* [25], paired with Light Detection and Ranging (LIDAR). Usually, this results in bulky and expensive systems requiring an appropriate space on-board the vehicle and consequently not being fit for small autonomous vessels. In [26], the proposed low-cost architecture bases its functioning on a standard color daylight camera, an uncooled infrared camera and a single laser rangefinder. The two cameras are devoted to determine two of the coordinates that characterize the position of the obstacles (azimuth and elevation), while the rangefinder is in charge of computing the distance component of the obstacle position. The basic idea consists in acquiring, as quickly as possible, a panoramic view of the 180 degree ahead the vehicle with the passive sensors and, in a second phase, using the laser rangefinder to inspect the possible obstacles identified.

Although the present work does not cater to USVs, the sensors employed in the aforementioned studies and the innovative systems that encompass them can be adapted to rovers and UAVs. Compactness and low cost are two features worth delving into.

2.2.2 Unmanned Ground Vehicles (UGV)

Unmanned Ground Vehicles (UGVs) play a pivotal role in a myriad of applications, including transport and logistics. To achieve better distribution efficiency, [27] proposed an intelligent actuator of an indoor logistics system by fusing multiple sensors. This actuator is based on a four-wheel differential chassis, equipped with sensors, including a Red Green Blue (RGB) camera, a LIDAR and an indoor inertial navigation system, by which autonomous driving can be realized. Multi-sensor fusion is used to recognize

the type and position of obstacles, which is the basis for collision avoidance and overtaking. A similar approach was used in [28], to account for obstacles that are too small, or that are invisible because they are outside the LIDAR's field of view (see Fig. 2.3(a)). In this case, a movable RGB-Depth (RGB-D) camera was mounted on the head of the humanoid robot R1, with the purpose of investigating active control strategies to effectively scan the environment. This work very clearly highlights the importance of sensor selection, placement and orientation.



(a) Obstacle that is not detected by a LIDAR on the robot base.

(b) LIDAR and RGB-D vision range of R1.

Figure 2.3: Obstacle detection on a mobile humanoid robot [28].

On the opposite side of the spectrum, [29] explores the obstacle avoidance capabilities of a blind walking hexapod robot. This robot is not equipped with proximity sensors and therefore exploits internal sensors to obtain information about its surroundings. These include an Inertial Measurement Unit (IMU) on the torso, and angle feedback and torque feedback at each joint. The proprioceptive signals from the hexapod robot are obtained through signal processing of the internal sensors. Proprioception is the perception of one's own space and position in three-dimensional space, allowing the body to perceive changes in limb movements. Ultimately, this study places emphasis on the potential of internal sensors for obstacle avoidance.

2.2.3 Unmanned Aerial Vehicles (UAV)

Since UAVs are the primary object of study of this work, Tab. 2.1 summarizes the most relevant characteristics of state-of-the-art S&A systems for this particular type of vehicle. Regarding obstacle avoidance, most research focuses on multirotor UAVs. Among these, solutions typically include cameras and LIDAR technology. However, that is not the case for the solution presented in [30], specifically designed to map indoor spaces with planar structures through graph optimization. Using many 1D lasers to maximize the orientations that are being covered, as opposed to using a single 2D LIDAR allows for a more accurate hypothesis of the planar structures, free of assumptions about the horizontal and vertical planes.

In [19], different detection systems for fixed-wing UAVs were simulated before reaching a potential optimal solution, including two laser rangefinders and a RADAR. Regarding the study conducted in [20], rather than focusing on optimization, the author opted for validating the performance of a S&A framework employing simpler rangefinders (sonar and laser).

Table 2.1: Sensor solutions deployed in S&A systems described in literature.

Reference	**Wing configuration**	**Sensors**
[30]	Multirotor	(6x) Laser Rangefinder
[31]	Multirotor	LIDAR
[32]	Multirotor	(3x) Depth Camera LIDAR
[33]	Multirotor	Depth Camera Tracking camera LIDAR
[34]	Fixed-Wing	(2x) Microphone
[19]	Fixed-Wing	(2x) Laser Rangefinder RADAR
[20]	Fixed-Wing	Ultrasonic Sensor Laser Rangefinder

2.3 Non-cooperative Obstacle Sensing

Non-cooperative sensors can be categorized as active or passive, depending on the source of the detected signal. While passive sensors rely on external signals, active sensing is based on analyzing signals that travelled from the source to the obstacle and go back to the source after being reflected [35].

The next subsections are dedicated to different models of active non-cooperative sensors: the Ultrasonic Sensor, the Laser Rangefinder, the LIDAR (Light Detection and Ranging) and the RADAR (Radio Detection and Ranging), as illustrated in Fig. 2.4, followed by their comparative analysis.

(a) Ultrasonic Sensor. (Source: MaxBotix.)

(b) Laser Rangefinder. (Source: LightWare.)

(c) LIDAR. (Source: LightWare.)

(d) RADAR. (Source: Ainstein.)

Figure 2.4: Active non-cooperative sensors.

2.3.1 Ultrasonic Sensor

One of the direct methods for determining the distance between a UAV and an obstacle is ultrasonic sensing. In much the same way as bats use echolocation to find prey, the sensor generates a sound, which is then reflected by the obstacle and recorded by the sensor. If the velocity of the radiated sound in the air medium is known, the distance to the object can be calculated. In other words, ultrasonic sensors rely on time-of-flight to measure distance and return a range. The range, however, is the distance from the point of greatest reflection to the obstacle, not the distance in a straight line [36].

Under dim lighting circumstances, such as smoke or fog, ultrasonic sensors are particularly reliable. Furthermore, not only is the system useful to detect transparent obstacles, but it is also a low-cost solution. However, because these are proximal sensors, their signal quickly attenuates and their capacity to measure distance is typically limited to less than 10 meters. Additionally, they are unable to detect sound-absorbing materials like clothing, which makes the technology unreliable for detecting people, for example. Because their simultaneous use addresses each sensor's shortcomings, it is fairly usual to combine this type of sensor with an infrared sensor [37]. Another limitation worth considering regards the speed of sound. If the system is not recalibrated for changes in temperature or air density, changes in the speed of sound lead to errors in distance measurements.

Ultrasonic wave propagation can be applied to act as an airflow velocity sensor for UAVs. In reference [38], transit-time flow meters act as an effective way to measure flow velocity in a clean fluid. The process consists of an ultrasonic signal being passed between two transducers, both upstream and

downstream of the oncoming fluid (see Fig. 2.5). From the time taken for the signal to travel upstream and downstream, the flow velocity can be calculated. Not only was this method validated, but the author also reached conclusions for the overall design, showing that it can produce accurate results which compare with those of current instrumentation sensors. Using ultrasonic sensors proves to be advantageous due to the ease at which this simple technology can be sized down. Thus, future work in this field is envisaged in the miniaturisation of the equipment, facilitating its installation on UAVs.

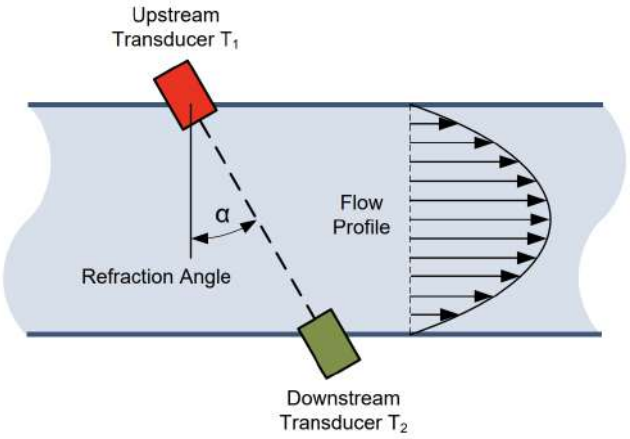


Figure 2.5: Operating principle of a transit-time flow meter. An ultrasonic wave is emitted alternatively by both transducers, T_1 and T_2 [38].

2.3.2 Laser Rangefinder

Laser rangefinders are able to compute distances to obstacles by emitting a laser pulse and measuring the time it takes for the reflected beam to be detected, given that laser light beams move at a known speed. This principle is quite common among sensors, accounting for lightweight, low-cost technology [39]. However, it is limited by weather conditions, as laser light might scatter in the presence of clouds, fog or atmospheric attenuation.

In [40], this sensor is included in a a comprehensive control and navigation scheme for an indoor UAV system (see Fig. 2.6). The laser rangefinder, capable of scanning a level plane, is used in addition to the common inertial measurement unit. This setup allows the UAV to estimate its own velocity and position robustly, while flying along the internal walls of a room and avoiding collision, without any remote sensory information or off-line computational power, i.e., after being issued the main navigation command, the UAV does not need to maintain any wireless link to the Ground Control Station.

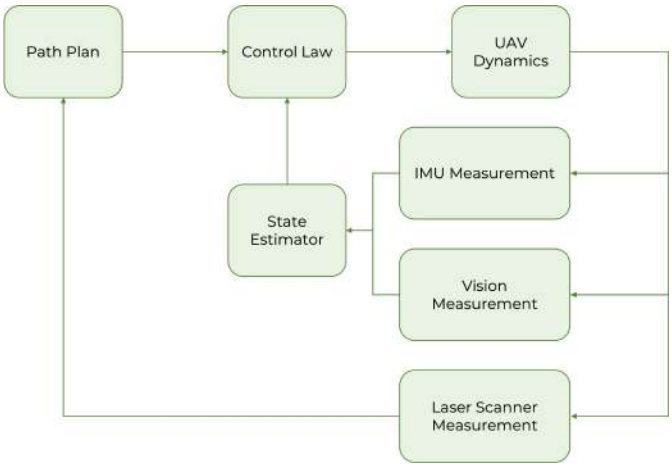


Figure 2.6: Indoor navigation system [40].

2.3.3 Light Detection and Ranging (LIDAR)

Light Detection and Ranging (LIDAR) emits short and precise laser light impulses with high frequency, that in turn, are reflected and received again by the sensor, measuring the time it took for them to return. LIDAR employs ultraviolet, visible, or near infrared light (electromagnetic wavelengths from 300 nm to 1000 nm) to scan the environment and reproduce it in digital 3-D images, such as the example in Fig. 2.7. Although this technology is similar to the Laser Rangefinder's, it is multidirectional. Thus, its execution goes beyond simply detecting an obstacle's range. A 3-D point cloud can be acquired through a vast array of distance measurements, simply by attaching a scanning surface (e.g. an electrical servo) to the laser or if an oscillating mirror deflects the laser beams.

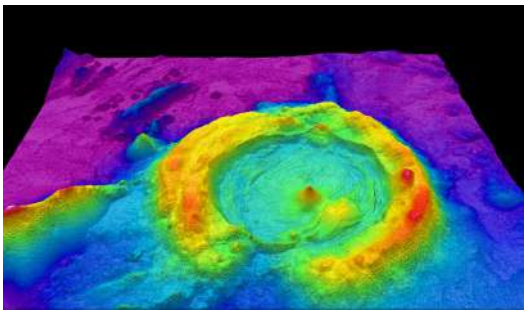


Figure 2.7: Airborne LIDAR Bathymetric Technology: High-resolution multibeam LIDAR map depicting seafloor geology, in shaded relief and coloured by depth (Source: NOAA Ocean Exploration & Research).

Knowing the exact instant of emission and detection of a light impulse allows for great accuracy in distance calculation. Nonetheless, due to the properties of light, LIDAR has a limited angular resolution (determined by a half-power beamwidth). This implies that a light beam traveling 10 km from its source will only encompass a circular area with a diameter of less than 5m. Thus, intelligent scanning techniques are needed to expedite this process.

As previously mentioned, in the presence of clouds and fog, laser light scatters. Moreover, air attenuation influences light, limiting the LIDAR's detectable range. A proper level of emitted power and frequency must be selected to avoid human eye hazard (in case a light beam penetrates through a cockpit). Due to these limitations, LIDAR should be incorporated into sensor fusion systems in low-speed airspace, rather than being employed as a standalone component of S&A systems [35].

Figure 2.7 depicts seafloor geology through LIDAR technology. Recently, however, UAV-based LIDAR data has also been used for tree measurements, seen in Fig.2.8. In [41], a LIDAR scanner was mounted on an eight-rotor UAV platform, operating at 903nm and transmitting 700000 pulses per second. The LIDAR data is used to generate models that represent the height of the mangroves, at four spatial resolutions. In turn, from the height models, individual mangroves are detected and crowns are delineated, using two different algorithms, and thus allowing the parameters to be extracted.

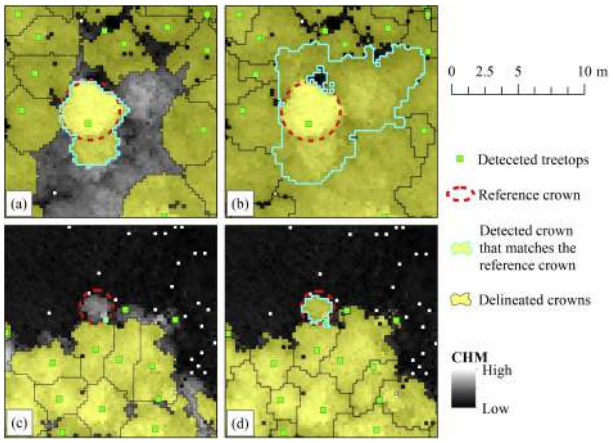


Figure 2.8: Individual mangrove tree measurements using UAV-based LIDAR data [41].

2.3.4 Radio Detection and Ranging (RADAR)

Radio Detection and Ranging (RADAR) is one of the most popular sensing technologies. It consists of a transmitting antenna producing electromagnetic waves (in the radio or microwave spectrum) and a receiving antenna, which collects waves echoed from static or dynamic obstacles [35]. By measuring the time lapse between the transmitted and received signal, it is possible to determine the distance between the sensor and the target, since radio waves move at a known speed, in a way that can be projected mathematically. Despite being very similar to the LIDAR, RADAR technology is distinguished by the frequency of the emitted radiation.

RADARs are considered the most comprehensive airborne surveillance sensing method, since its technology is reliable at day or nighttime, in all-weather conditions. In collision detection, a Monostatic Pulse RADAR is most commonly used, although emitting a continuous wave (CW) is also possible. The Continuous Wave RADAR's dimensions can be suitable for small UAVs [35].

However, proper detection of targets can become difficult when dealing with scenes having a large amount of targets, due to challenges in initiation and maintenance of tracks. Also, detection range requirements to enable safe conflict detection and avoidance may be hard to fulfill due to small Radar

Cross Section (RCS) of targets such as other micro-UAS. Nevertheless, probability of detection can be increased by installing higher gain antennas. In turn, it negatively impacts angular coverage and might lead to the need of multiple antennas or scanning systems. Fusion of data from multiple sensors, both radar and vision-based, could be an effective way to enhance S&A capabilities [42]. Additionally, a trade-off in low-power and long-range is expected from an effective S&A RADAR system for UAVs.

Reference [43] explores anti-personnel landmine detection by using a UAV in combination with a Ground Penetrating Synthetic Aperture RADAR (GPSAR). The system, shown in Fig. 2.9, aims at accelerating the process of land release in humanitarian de-mining. After being detected by the RADAR, suspicious objects are marked for further investigations using different sensor principles. The Ground Penetrating RADAR module includes a 1 to 4 GHz side-looking Frequency Modulated Continuous Wave (FMCW) RADAR. This frequency was selected as a compromise between penetration depth, resolution, and geometrical dimensions suitable for UAV applications. The module is also equipped with a RADAR and LIDAR altimeter, and a Real Time Kinematic Global Navigation Satellite System (RTK GNSS). The image processing is done offline using a back-projection algorithm.

Figure 2.9: Measurement system with a GPSAR sensor module mounted below the UAV [43].

2.3.5 Sensors' Comparative Analysis

To propose a S&A system, the devices to be used for sensing need to be determined. It became clear from these models' brief explanation that while ultrasonic sensors are more cost-effective, Laser Rangefinders, LIDARs and RADARs are better suited for longer distances. For a better understanding of the sensors' advantages and faults, these and other properties are listed in Tab. 2.2.

Table 2.2: Sensors qualitative comparison [19].

Sensor	Weight	Electric Power	Signal Processing	Cost	Range	Directionality	FOV
Ultrasonic Sensors	low	low	simple	low	low	directional	medium
Laser Rangefinder	low	low	simple	medium	high	directional	very narrow
LIDAR	medium	medium	complex	high	high	multidirectional	very narrow
RADAR	medium	low	complex	high	high	(multi)directional	broad

Table 2.2 shows that laser and sonar sensors have complementary properties. Sonars can only assess distances up to 10 m due to substantial signal attenuation, whereas laser sensors have a significant

range. Additionally, laser sensors struggle in low-light situations and ultrasonic sensors are unable to detect sound-absorbing materials, but their simultaneous use can overcome these shortcomings. These are also far less expensive than other sensors, while still being reliable and quite simple to use. Regular laser rangefinders have a limited field of view (FOV), which limits their ability to identify objects that are not directly in front of them. A LIDAR, which consists of a multidirectional laser rangefinder, can be utilized to address this problem. However, since an electrical servo or oscillating mirror system must also be installed, these represent an increase in cost, weight, and required power.

RADARs can be directional or multidirectional, since they can either comprise a patch or a scanning antenna. Some of the advantages of this technology include offering a wide FOV and a range that is comparable to a laser rangefinder, simultaneously being power-efficient. The key distinction between this technology and that employed by laser rangefinders and LIDARs is the frequency of the radiation that is emitted: RADARs emit 24 GHz microwave radiation, whereas lasers use 300 THz infrared light.

2.4 Collision Avoidance Algorithms

To aid in collision avoidance and path replanning, some algorithms must be implemented. This section is composed by theoretical expositions of methods that are typically applied in flight controller software.

In order to avoid obstacles, the UAV attitude and position must be known. Thus, it is proposed to resort to the Extended Kalman Filter (EKF). Once the controller is aware of the UAV states, the Vector Field Histogram (VFH) and Potential Fields methods guarantee obstacle avoidance and its redirection. Although these methods are explored in the present work, other possible avoidance algorithms include D* Lite [44] and the Genetic Algorithm [45]. These algorithms are to be implemented in the flight controller software, which is outside the scope of this work.

2.4.1 Extended Kalman Filter (EKF)

Flight control software is commonly equipped with a sophisticated attitude and position estimation system, known as the Extended Kalman Filter (EKF) algorithm. Aided by rate gyroscopes, accelerometer, compass (magnetometer), Global Positioning System (GPS), airspeed and barometric pressure measurements, EFK can estimate vehicle position, velocity and angular orientation.

The following steps describe the filter's working [46]:

1. IMU angular rates are integrated to calculate the angular position;
2. IMU accelerations are converted using the angular position from body X, Y, Z to earth North, East and Down axes and corrected for gravity;
3. Accelerations are integrated to calculate the velocity;
4. Velocity is integrated to calculate the position;

This process, from (1) to a (4), is referred to as 'State Prediction'. A 'state' is a variable we are trying to estimate like roll, pitch, yaw, height, wind speed, etc. The filter has other states besides position, velocity and angles that are assumed to change slowly. These include gyro biases, Z accelerometer bias, wind velocities, compass biases and the earth's magnetic field. These other states are not modified directly by the 'State Prediction' step but can be modified by measurements as described later;

5. Estimated gyro and accelerometer noise are used to estimate the growth in error in the angles, velocities and position calculated using IMU data. Making these parameters larger causes the filters error estimate to grow faster. If no corrections are made using other measurements (e.g. GPS), this error estimate will continue to grow. These estimated errors are captured in a large matrix called the 'State Covariance Matrix';

   Steps (1) to (5) are repeated every time we get new IMU data until a new measurement from another sensor is available.

   If we had a perfect initial estimate, perfect IMU measurements and perfect calculations, then we could keep repeating (1) to (4) throughout the flight with no other calculations required. However, errors in the initial values, errors in the IMU measurements and rounding errors in our calculations mean that we can only go for a few seconds before the velocity and position errors become too large.

   The Extended Kalman Filter algorithm provides us with a way of combining or fusing data from the IMU, GPS, compass, airspeed, barometer and other sensors to calculate a more accurate and reliable estimate of our position, velocity and angular orientation.

   The following example describes how GPS horizontal position measurements are used, however the same principal applies to other measurement types (barometric altitude, GPS velocity, etc).

6. When a GPS measurement arrives, the filter calculates the difference between the predicted position from (4) and the position from the GPS. This difference is called an 'Innovation';

7. The 'Innovation' from (6), 'State Covariance Matrix' from (5), and the GPS measurement error are combined to calculate a correction to each of the filter states. This is referred to as a 'State Correction';

   This is the clever part of the a Kalman Filter, as it is able to use knowledge of the correlation between different errors and different states to correct states other than the one being measured. For example GPS position measurements are able to correct errors in position, velocity, angles and gyro bias.

   The amount of correction is controlled by the assumed ratio of the error in the states to the error in the measurements. This means if the filter thinks its own calculated position is more accurate than the GPS measurement, then the correction from the GPS measurement will be smaller. If it thinks

its own calculated position is less accurate than the GPS measurement, then the correction from the GPS measurement will be larger.

8. Because we have now taken a measurement, the amount of uncertainty in each of the states that have been updated is reduced. The filter calculates the reduction in uncertainty due to the 'State Correction', updates the 'State Covariance Matrix' and returns to step (1).

2.4.2 Vector Field Histogram (VFH)

The Vector Field Histogram algorithm is a method for mobile robot obstacle avoidance. Some enhanced versions of this method (VFH+, VFH*, 3DVFH) are used in flight control software and thus will be further explored.

VFH+

The VFH+ method builds a polar histogram around the robot's current position, looks for openings in the histogram, and then determines between one and three suitable directions for each opening. VFH+ assigns a cost value to each of these primary candidate directions and then selects the primary candidate direction with the lowest cost as its new direction of motion [47].

Primary Polar Histogram

The first data reduction stage maps the active region C_a of the map grid C onto the primary polar histogram H^p. The active region C_a is a circular window of diameter w_s that moves with the robot. The content of each active cell in the map grid is treated as an obstacle vector. The vector direction $\beta_{i,j}$ is determined by the direction from the active cell to the robot center point (RCP),

$$\beta_{i,j} = \arctan\left(\frac{y_j - y_o}{x_i - x_o}\right), \tag{2.1}$$

where x_0, y_0 are the present coordinates of the RCP and x_i, y_j are the oordinates of active cell $C_{i,j}$.

The vector magnitude $m_{i,j}$ for an active cell $C_{i,j}$ is given by

$$m_{i.j} = c_{i,j}^2\left(a - bd_{i,j}^2\right), \tag{2.2}$$

where $c_{i,j}$ is the certainty value of active cell $C_{i,j}$ and $d_{i,j}$ is the distance from active cell $C_{i,j}$ to the RCP.

For $m_{i,j}$ to be equal to the square of $c_{i,j}$ at the boundary of the active region, the parameters a and b are chosen according to

$$a - b\left(\frac{w_s - 1}{2}\right)^2 = 1. \tag{2.3}$$

Based on the obstacle vectors, the primary polar histogram H^p is built. H^p has an arbitrary angular resolution α so that $s = 360^\circ/\alpha$ is an integer. Each angular sector k corresponds to a discrete angle $\varphi = k \times \alpha$.

The VFH $+$ method uses an analytically determined low-pass filter to compensate for the width of the robot. Obstacle cells in the map are enlarged by the robot radius r_r, which is defined as the distance from the robot center to its furthest perimeter point. For further safety, the obstacle cells are actually enlarged by a radius $r_{r+s} = r_r + d_s$ where d_s is the minimum distance between the robot and an obstacle.

Binary Polar Histogram

Based on the primary polar histogram H^p and two thresholds, τ_{low} and τ_{high} , a binary polar histogram H^b is built. Instead of having polar density values, the sectors of H^b are either free (0) or blocked (1). The binary polar histogram indicates which directions are free for a robot that can instantaneously change its direction of motion. At time n, the binary polar histogram is updated by the following rules:

$$H^b_{k,n} = \begin{cases} 1 & \text{if } H^p_{k,n} > \tau_{\text{high}} \\ 0 & \text{if } H^p_{k,n} < \tau_{low} \\ H^b_{k,n-1} & \text{otherwise.} \end{cases} \tag{2.4}$$

Masked Polar Histogram

The VFH+ method uses a simple approximation of the trajectory of most mobile robots. It assumes that the robot's trajectory is based on circular arcs (constant curvature curves) and straight lines. The values of the minimum steering radii as a function of the robot velocity can easily be measured. These radii are defined for both sides as $r_r = 1/k_r$ and $r_l = 1/k_l$. With these parameters and the map grid, it is possible to determine which additional sectors are blocked by obstacles because of the robot trajectory. To take the width of the robot into account again, the obstacles are enlarged by r_{r+s}. If a trajectory circle and an enlarged obstacle cell overlap, all directions from the obstacle to the direction opposite the robot's heading are blocked. An example can be seen in Fig. 2.10.

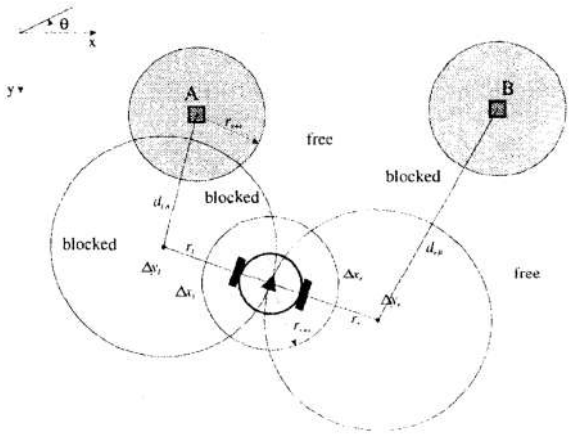


Figure 2.10: Example of blocked directions [48].

The positions of the right and left trajectory centers relative to the current robot position are respectively defined by

$$\Delta x_r = r_r \cdot \sin\theta \quad \Delta y_r = r_r \cdot \cos\theta, \text{ and} \tag{2.5}$$

$$\Delta x_l = -r_l \cdot \sin\theta \quad \Delta y_l = -r_l \cdot \cos\theta. \tag{2.6}$$

The distances from an active cell $C_{i,j}$ to the right and left trajectory centers are respectively given by

$$d_r^2 = (\Delta x_r - \Delta x(j))^2 + (\Delta y_r - \Delta y(i))^2, \text{ and} \tag{2.7}$$

$$d_l^2 = (\Delta x_l - \Delta x(j))^2 + (\Delta y_l - \Delta y(i))^2. \tag{2.8}$$

An obstacle cell blocks the directions to its right if

$$d_r^2 < (r_r + r_{r+s})^2 \quad \text{[condition 1]} \tag{2.9}$$

And an obstacle cell blocks the directions to its left if

$$d_l^2 < (r_i + r_{r+s})^2 \quad \text{[condition 2]} \tag{2.10}$$

By checking every active cell with these two conditions, we obtain two limit angles: φ_r for right angles and φ_l for left angles. We also define $\varphi_b = \theta + \pi$ as the direction opposite to the current direction of motion.

This stage can be implemented very efficiently by an algorithm that only considers cells that have an influence on either φ_r or φ_l:

1. Determine φ_b. Set φ_r and φ_l equal to φ_b;
2. For every cell $C_{i,j}$ in the active window C_a with $c_{i,j} > \tau$:
   (a) If $\beta_{i,j}$ is to the right of θ and to the left of φ_r, check condition 1 . If condition is satisfied, set φ_r equal to $\beta_{i,j}$;
   (b) If $\beta_{i,j}$ is to the left of θ and to the right of φ_l, check condition 2 . If condition is satisfied, set φ_l equal to $\beta_{i,j}$.

With φ_r, φ_l, and the binary polar histogram, it is possible to build the masked polar histogram:

$$H_k^m = \begin{cases} 0 & \text{if} \quad H_k^b = 0 \text{ and } (k \cdot \alpha) \in \{[\varphi_r, \theta], [\theta, \varphi_l]\} \\ 1 & \text{otherwise.} \end{cases} \tag{2.11}$$

The masked polar histogram shows which directions of motion are possible at the current speed. If all sectors were blocked, the robot could not proceed at the current speed, and it would have to determine a set of new values (φ_r, φ_l) based on a slower speed. If the masked polar histogram were still blocked in all directions, the robot would have to stop immediately.

Selection of the Steering Direction

The masked polar histogram shows which directions are free of obstacles and which ones are blocked. However, some free directions are better candidates than others for the new direction of motion. The VFH+ method first finds all openings in the masked polar histogram and then determines a set of possible candidate directions. The candidate direction k_d with the lowest cost is then chosen to be the new direction of motion $\varphi_d = k_d \times a$.

In the first step, the right and left borders k_r and k_l of all openings in the masked polar histogram are determined. Two types of openings are distinguished: wide and narrow. An opening is considered wide if the difference between its two borders is larger than $s_{\max}$ sectors. Otherwise, the opening is considered narrow.

For a narrow opening, there is only one candidate direction so that the robot steers through the center of the gap between the corresponding obstacles:

$$c_d = \frac{k_r + k_l}{2} \quad \text{centered direction } ^+ \tag{2.12}$$

For a wide opening, there are at least two candidate directions, c_r to the right and c_l to the left side of the opening. The target direction is also a candidate direction, if it lies between the two other candidate directions:

$$\begin{aligned} c_r &= k_r + \frac{s_{\max}}{2} \quad && \text{towards the right side} \\ c_l &= k_l - \frac{s_{\max}}{2} \quad && \text{towards the left side} \\ c_l &= k_l && \text{if } k_t \in [c_r, c_l] \end{aligned} \tag{2.13}$$

The candidate directions c_r and c_l make the robot follow an obstacle contour at a safe distance, while c_t leads the robot towards the target direction.

The cost function g is defined as a function of a candidate direction c,

$$g(c) = \mu_1 \cdot \Delta\left(c, k_t\right) + \mu_2 \cdot \Delta\left(c, \frac{\theta_n}{\alpha}\right) + \mu_3 \cdot \Delta\left(c, k_{d,n-1}\right), \tag{2.14}$$

where $\Delta\left(c_l, c_2\right)$ is a function that computes the absolute angle difference between two sectors c_1 and c_2 so that the result is $\leq s/2$. In short, the first term is responsible for the goal- oriented behavior, while the second and third term make the mobile robot commit to a direction.

The higher μ_1 is, the more goal-oriented the robot's behavior. The higher μ_2 is, the more the robot tries to execute a smooth path with a minimum change of direction of motion. The higher μ_3 is, the more the robot tries to head towards the previously selected direction and the smoother are the motor commands. Only the relationship between the three parameters is important, not their magnitudes. To guarantee a goal oriented behavior, the following condition must be satisfied:

$$\mu_1 > \mu_2 + \mu_3. \quad \text{[condition 3]} \tag{2.15}$$

Further details on the VFH+ algorithm may be found in [48].

VFH*

In contrast to VFH+, VFH* analyzes the consequences of heading towards each primary candidate direction before making a final choice for the new direction of motion. For each primary candidate direction, VFH* computes the new position and orientation that the robot would have after moving for a projected step distance d_s. At every projected position, VFH+ is again used to construct a new polar histogram based on the map information. This histogram is then analyzed for candidate directions. By repeating this process n_g times, a search tree of depth n_g is built, where the end nodes (goals) correspond to a total projected distance of $d_t = n_g \times d_s$.

The goal depth n_g is proportional to the total projected distance d_t. The higher d_t is selected, the larger the total look-ahead, and the better the results of VFH* are. However, if this parameter is selected too high, the obstacle avoidance algorithm is slowed down substantially. If possible, this parameter should be set close to the range of the robot's sensors. It is important to note that the VFH+ method is a special case of the VFH* method, where n_g is set equal to one, such that $d_t = d_s$.

The goal of this search process is to find a suitable projected trajectory of distance d_t. Nodes in the search tree represent the projected positions and orientations of the mobile robot. Arcs represent the candidate directions leading from one position to another.

The first steps: building the polar histogram and determining the corresponding candidate directions, are performed in the same way as in the VFH+ algorithm. The cost associated with a node is simply the sum of the costs of the branches leading back to the start node. The primary candidate direction that leads to the end node with the smallest total cost is then selected as the new direction of heading φ_d[47].

3DVFH+

The 3DVFH+ algorithm is based on the two-dimensional VFH+ algorithm. The algorithm uses an octomap to determine where the obstacles are located, making a 2D Primary Polar Histogram (see Fig. 2.11). Taking into account the physical capability of the robot, the algorithm will find multiple paths, give them a path weight and determine the path with the lowest path weight. This path is then used to calculate a 3D motion for the robot [49].

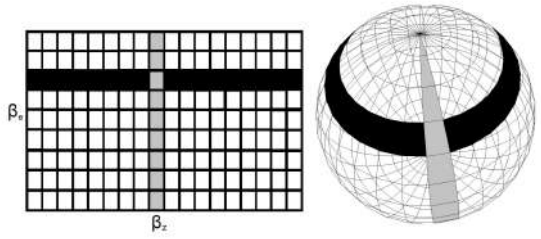


Figure 2.11: 2D Polar histogram [49].

2.4.3 Potential Fields

The Potential Fields algorithm was the chosen approach to solve the local path planning problem in [19] [50]. Therefore, this method was also employed in the present work's simulations.

To adapt this algorithm to the present work, a comprehensive definition must be established, wherein each detected obstacle is associated with various safety zones, each in turn serving pivotal roles in both collision detection and avoidance simulations. The obstacles can be modeled as spheres, as represented in Fig. 2.12. The collision radius (R_c) demarks the obstacle's volume, thus, a collision event is registered if the UAV trespasses this radius. The safety radius (R_s) specifies the minimum separation distance between the UAV and the obstacle, accounting for potential deviations and uncertainties that may arise during detection and path prediction phases. When breached, a close call is registered. Finally, the detection radius (R_d) limits the range from which an obstacle is considered by this algorithm. The selection of appropriate values for these radii is contingent upon specific parameters, such as the UAV's dimensions, velocity, and the approach type towards obstacles: the R_c should closely match the UAV's size, while the R_s ought to be slightly longer and R_d should align with the range of the sensors employed.

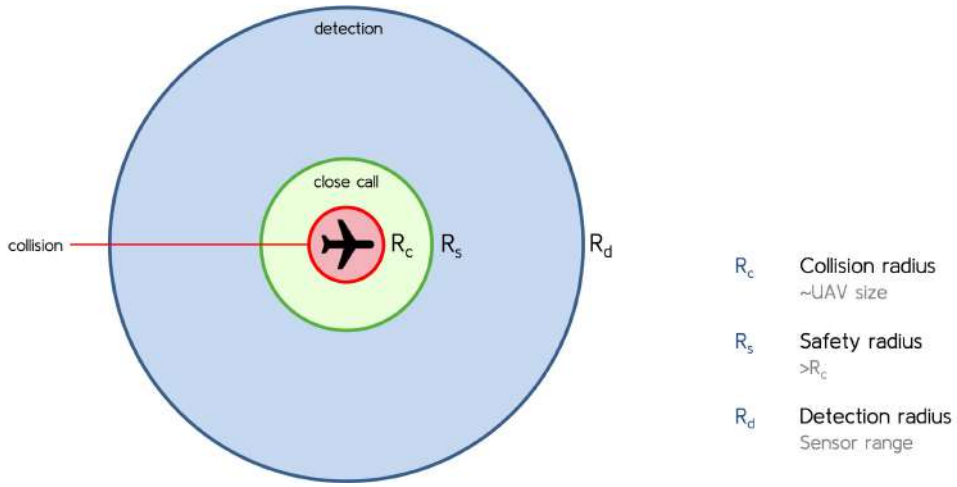


Figure 2.12: Safety radii.

Based on Coulombs law, the Potential Fields method conceptualizes waypoints and obstacles as charged particles. Within this analogy, waypoints create an attractive field, whereas obstacles create a repulsive field and the sum of all forces is used to generate the direction of motion, visually represented in Fig. 2.13. Upon successfull implementation, this will allow the simulated UAV to avoid the obstacles' collision and safety radii by replanning its trajectory, as shown in Fig. 2.14.

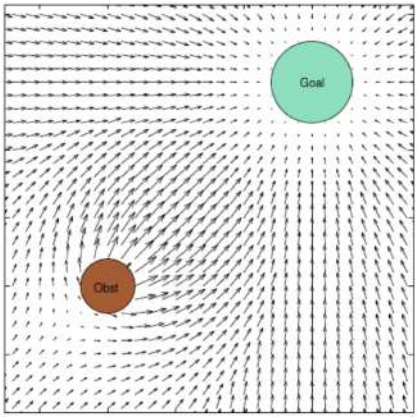


Figure 2.13: Potential field representation [51].

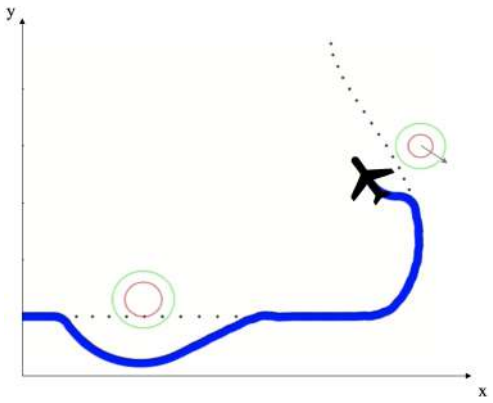


Figure 2.14: Path replanning.

Attractive potential

The attractive potential is given by

$$\mathbf{f}_{at} = \alpha_{PF} \frac{P_{\text{close}} - P}{\|P_{\text{close}} - P\|} + (1 - \alpha_{PF}) \frac{P_{\text{next}} - P_{\text{close}}}{\|P_{\text{next}} - P_{\text{close}}\|}, \tag{2.16}$$

where the first term is responsible for guiding the UAV to the nearest point of the global path and the second term is responsible for guiding the UAV to the next defined waypoint. P is the UAV's position and P_{close} is the closest point of the global path. Subtracting both positions and dividing by its norm results in a unit vector pointing from the UAV to the closest point of the path. The second term functions under the same principles, where P_{next} is the position of the next waypoint. The α_{PF} term is responsible for giving more or less predominance to each term.

Repulsive potential

Using a simple repulsive potential to avoid obstacles is not feasible since that would lead to irregular motion around the obstacle. Adding a swirling motion to the potential flow solves this problem and makes the evasion start sooner, since the UAV will evade the obstacle instead of just keeping the distance to the obstacle.

The swirling term is given by

$$\mathbf{s}_{dir} = \frac{\hat{\mathbf{k}} \times \mathbf{d}_0}{\|\mathbf{d}_0\|}, \tag{2.17}$$

where $\hat{\mathbf{k}}$ is the unit vector in the z direction $(0, 0, 1)$ and $\mathbf{d}_0$ is the vector pointing from the obstacle to the UAV. Computing $\mathbf{s}_{\text{dir}}$ this way results in a vector that makes the UAV circumvent the obstacle counterclockwise. When the obstacle needs to be circled in the other direction, the symmetric of $\mathbf{s}_{dir}$ is used.

To avoid the UAV being trapped around the obstacle, the generated field needs to become zero once the obstacle is overcome. To check this condition, an angle θ between the desired direction of motion (m) and the direction of the obstacle needs to be computed from

$$\theta = \arccos\left(\frac{\mathbf{m}\cdot\mathbf{d}_0}{\|\mathbf{m}\|\,\|\,\mathbf{d}_0\|}\right). \tag{2.18}$$

If this angle is larger than a $\theta_{\text{cut-off}}$, the obstacle will have been overcome, making the field null. Knowing this, the potential associated to the obstacle is described by

$$\mathbf{f}_{\text{rep}} = \begin{cases} \infty\frac{\mathbf{d}_0}{\|\mathbf{d}_0\|} & \text{, if } \|\mathbf{d}_0\| \leq R_c \\ S_{\max}\mathbf{s}_{dir} & \text{, if } R_c < \|\mathbf{d}_0\| \leq R_s \\ S_{\max}\frac{R_a-\|\mathbf{d}_0\|}{R_a-R_s}\mathbf{s}_{dir} & \text{, if } R_s < \|\mathbf{d}_0\| \leq R_a \\ 0 & \text{, if } \|\mathbf{d}_0\| \geq R_a \vee \theta \leq \theta_{cut-off} \end{cases} \tag{2.19}$$

This field is different according to the distance between the obstacle and the UAV, as shown in Fig. 2.15. When the UAV is within the collision zone, the field exerts an infinite repulsive force. Within the safety zone, the field aligns with the $\mathbf{s}_{dir}$, as previously defined, and has an intensity of S_{max}, a constant proportional to the UAV's velocity. Within the action zone, the field closely resembles that of the safety zone, yet incorporates an additional gradient term ensuring that the field's intensity linearly diminishes with increasing distance from the obstacle until it reaches null intensity at a distance of $\|\mathbf{d}_0\| = R_a$. Beyond the action zone, the obstacle's presence ceases to affect the UAV's motion, resulting in a null field intensity.

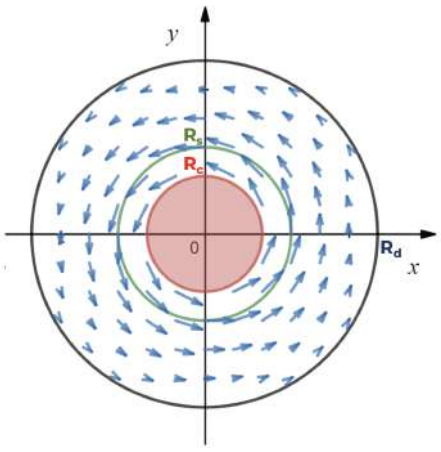
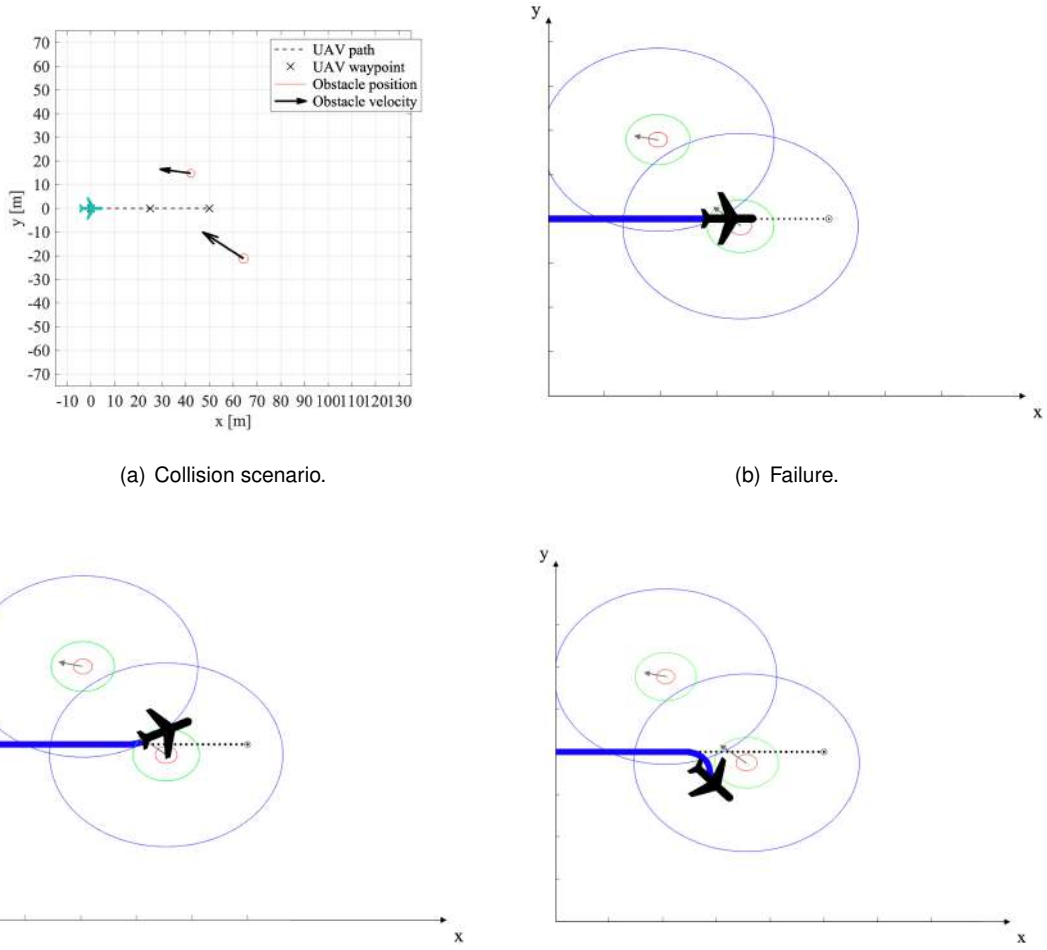


Figure 2.15: Repulsive field of an obstacle [20].

The Potential Fields algorithm used in this work ultimately translates to scenarios where a UAV with a pre-determined path follows way-points charged with attractive potential and attempts to stay away from obstacles (spheres) charged with repulsive potencial. For the same collision scenario (see details for its generation in Sec. 4.1), there are three possible outcomes, represented in Fig. 2.16: failure, close call and success. These outcomes depend on the sensor solutions in use, which is further explained in Chaps. 3 and 4.



(a) Collision scenario.

(b) Failure.

(c) Close call.

(d) Success.

Figure 2.16: Possible outcomes of a collision scenario.

2.4.4 Flight Controller Software

Currently, aircraft design strongly relies on automatic control systems to monitor and control its subsystems. These control systems can also provide artificial stability to improve the flying qualities of an aircraft. The above concepts are also applicable to UAVs [52], with the autopilot as its control element. The autopilot's task is to record values from the sensors, recalculate them and then actuate on the aircraft to meet the required trajectory [53]. Ardupilot and PX4 are the leading open source autopilot systems designed to control any type of unmanned vehicles, including fixed-wing aircraft and various rotary-wing platforms [54].

Figure 2.17: QGroundControl interface [55].

Figure 2.18: Mission Planner interface [56].

PX4 is an open source flight control software for drones and other unmanned vehicles. Its system architecture allows to modify the flight stack and middleware to add new flight modes and new air frames. PX4 supports a myriad of different sensors and actuators. QGroundControl provides full flight control and mission planning for any MAVLink enabled drone [55].

In much the same way as QGroundControl applies to PX4, Mission Planner is a full-featured ground station application for the ArduPilot open source autopilot project. It can be used can be used as a configuration utility or as a dynamic control supplement for a UAV.

Both controllers use the EKF as a state estimator. The forementioned collision avoidance algorithms (VFH and Potencial Fields) are also usually implemented. These algorithms can be modelled in order to be implemented and used for simulations in MATLAB.

The choice between PX4 and ArduPilot largely depends on specific requirements of the vehicle application, goals, and the platform to be used. PX4 has a strong and active development community, which often leads to the integration of newer technologies and features as they become available. In this work, a collision avoidance system is being projected for a UAV, therefore, using firmware that guarantees access to cutting-edge developments that can cater to this system is desirable. PX4 is also known for its advanced flight control capabilities. When working on a UAV that requires precise and sophisticated control, such as drones for mapping or applications that involve complex maneuvers, PX4 is a preferable choice. It uses a Real-Time Operating System (RTOS), which provides precise timing and control, making PX4 suitable for applications where timing and latency are crucial. Lastly, it offers more flexibility for customization. Considering this, PX4 was the chosen firmware for the initial implementation of the system and code development (see Chap. 6).

On the other hand, ArduPilot is known for its versatility and is suitable for a wider range of vehicles, including rovers. It offers extensive support for autonomous navigation and missions, making it a strong contender for rovers and ground-based applications where route planning and obstacle avoidance are important. ArduPilot can also be easily integrated with various sensors. On these grounds, Ardupilot was the chosen firmware for rover testing (see Sec. 7.2).

Chapter 3

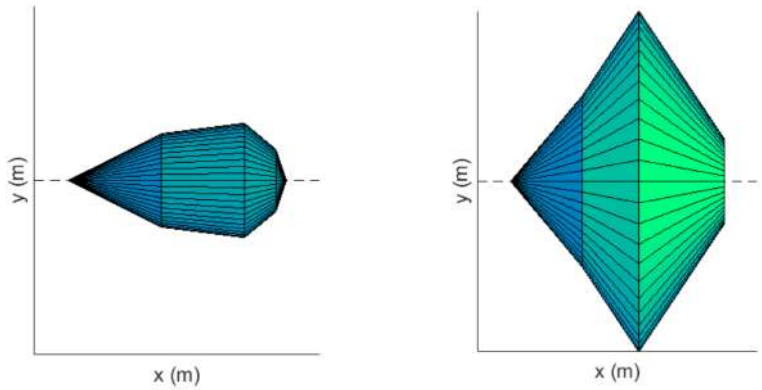
Sensor Modelling

A sensor model is an abstraction of the actual sensing process that specifies the data that a sensor may produce, how this data is constrained by the environment and how it can be improved by information gathered from other sensors.

Different models were developed by [50] and further adapted in the present work. In order to achieve the combination of sensors that guarantees the most fitting S&A results, the sensors' behaviour was compared through these models. These models were then incorporated in the obstacle detection and avoidance simulation environments.

3.1 Ultrasonic Sensor

Considering the ultrasonic sensor has a wide FOV, [20] differentiated between two different types of beam patterns: narrow and wide. Both beam patterns types have axial symmetry (see Fig.3.1) and are based on MaxBotix sonar models available on the market [57].



(a) Narrow beam. (b) Wide beam.

Figure 3.1: Ultrasonic sensor beam patterns.

Since the ultrasonic sensor only outputs a distance, it leaves all the interior beam points located at a specific distance from the UAV as potential object positions. This results in errors that shall be avoided,

as well as other issues that arise from sound reflection. The sound reflection law states that the reflected sound wave's angle with the normal of the surface is preserved. Thus, the ultrasonic sensor requires a perpendicular surface in order to detect an object, which in turn implies that the targets format is crucial to the mission's success. It is vital to recognize the final results can only be used as a reference given that these simulations only use spherical-shaped targets (and different target formats could either improve or worsen the outcome). In short, the model must check for these possibilities at all times:

1. The presence of any spherical surface point within the sonar beam pattern;

2. The perpendicularity of the sound wave direction with its reflecting surface.

Verifying these conditions requires considerable computing time. Therefore, a progressively complex approach that avoids unnecessary blocks of code was implemented [20]. First, the beam pattern is reduced to a cylinder. When the center of the obstacle is found to be inside the cylinder, a more thorough analysis is performed to identify which portion of the spherical surface, if any, is in fact inside the beam pattern.The last stage addresses the perpendicularity issue. The final surface computed in the preceding phase is defined as a list of points with a sampling ratio 25 times bigger for each spherical coordinate (in relation to the first list of points).

3.2 Laser Rangefinder

Given that all sensors' models may be implemented at an angle β relative to the longitudinal axis, this model assumes the use of two symmetrical sensors at the angles β and $-\beta$ whenever $\beta \neq 0$.

Considering the obstacles as spheres, this can be modeled as a simple interception between a line and a spherical surface. These can be given by

$$\begin{cases} \|\mathbf{x} - \mathbf{c}\|^2 = r^2 \\ \mathbf{x} = \mathbf{o} + d\hat{\mathbf{u}} \end{cases}, \tag{3.1}$$

where x is a generic point on the line and/or sphere, c is the centre point of the sphere, r is its radius, $\hat{\mathbf{u}}$ is the unit vector that defines the line direction in $3D$ space and d is the distance from the origin of the line. Combining both equations leads to an easily solvable quadratic equation,

$$d^2(\hat{\mathbf{u}} \cdot \hat{\mathbf{u}}) + 2d[\hat{\mathbf{u}} \cdot (\mathbf{o} - \mathbf{c})] + (\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - r^2 = 0. \tag{3.2}$$

After solving this equation, the model returns a solution if $0 < d_{\text{sol}} < \mathbf{R_d}$. In real conditions, the laser would not reach the furthest point, reflecting on the closest one. Therefore, if there are two solutions in this interval, only the smallest one prevails. The reflection point with the spherical surface can be easily obtained going back to the line equation $\mathbf{x}_{\text{sol}} = \mathbf{o} + d_{sol}\hat{\mathbf{u}}$ (Eq. 3.1).

3.3 LIDAR

The LIDAR model is very similar to the laser rangefinder's. As such, only the points that are closest to the sensor are detected. This implies that if an object is totally visible, it is considered that its half was detected and the remaining of the obstacle is reconstructed assuming symmetry, where the center of symmetry is the medium point of the segment connecting the first and last point of the cluster. In the present simulations, this distance corresponds to the diameter of the obstacle. This model discards obstacles that are hidden or outside FOV.

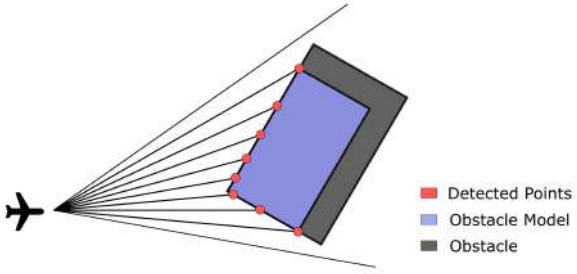


Figure 3.2: Obstacle reconstruction using a LIDAR [50].

A common issue lies within the higher distance between consecutive points in farther obstacles which results in smaller detected dimensions (see Fig. 3.2). To solve this problem, the measured diameter is passed through the time filter,

$$D_k = D_{k-1} + G\left(D_{\mathsf{m}} - D_{k-1}\right), \tag{3.3}$$

where $G(0 < G < 1)$ is the filter gain, D_k is the filtered diameter at instant t_k, D_{k-1} is the filtered diameter at instant t_{k-1}, and D_{m} is the measured dimension at instant t_k. The gain must be carefully chosen because it affects how quickly the dimensions change. While a small gain (*i.e.*, slow variation) is better for noisy surroundings, it is not appropriate for objects with high relative speeds. The gain is given by

$$G = 1 - \sqrt[n]{1-p}, \tag{3.4}$$

where p corresponds to a fraction that represents the desired accuracy of the dimensions and n corresponds to the number of filter cycles required to get an accuracy of p.

Classic Kalman filters were employed for the tracking phase, where the motion of obstacles was assumed to be two-dimensional, linear, and constant over successive scans. This simplification, which takes into account a high scanning frequency, accurately captures the targets' state.

3.4 RADAR

The RADAR sensor was modeled in the context of the Sense and Avoid system. *I.e.*, rather than being a lower-level model that would deal with signal and environment modeling, this model addresses the angular accuracy, update rate, range and FOV.

The state estimation is more complex than the one employed in the LIDAR model, given the RADAR sensor provides the range, bearing, and elevation of the observed obstacles. These outputs are polar, while the intruder dynamics are best described in rectangular coordinates. Due to its straightforward implementation, the converted measurement Kalman filter (CMKF) was chosen [50]. The 2-D model used in the simulations shown is represented by the equations below.

The compensation of the bias is multiplicative due to the use of the unbiased conversion and modeling the measurement errors as Gaussian white noise. The conversion can be calculated as

$$\begin{cases} x_m^u &= \lambda_\alpha^{-1} r_m \cos(\alpha_m) \\ y_m^u &= \lambda_\alpha^{-1} r_m \sin(\alpha_m) \end{cases} \tag{3.5}$$

where (x_m^u, y_m^u) are the measurements converted to the Cartesian frame, r_m is the measured range, α_m is the measured azimuth and λ_α is the bias compensation factor expressed as

$$\lambda_\alpha = e^{-\sigma_\alpha^2/2}, \tag{3.6}$$

where σ_α is the standard deviation of the noise in the azimuth measurements. The covariance matrix used in the Kalman Filter is given by

$$\mathbf{R}_u = \begin{bmatrix} \mathrm{var}(x_m^u \mid r_m, \alpha_m) & \mathrm{cov}(x_m^u, y_m^u \mid r_m, \alpha_m) \\ \mathrm{cov}(x_m^u, y_m^u \mid r_m, \alpha_m) & \mathrm{var}(y_m^u \mid r_m, \alpha_m) \end{bmatrix}, \tag{3.7}$$

with the details of the computation of these variances found in [58].

3.5 Multi-Sensor Data Fusion

All of these sensors (and respective models) provide input that allow the avoidance system to actuate. However, if the system's architecture is composed by more than a single sensor, the data provided must be merged in some way. Following best practices [59], the weighted filter method is used in the present study.

The principle behind this method is simple: each sensor is given a weight that is based on how reliable it is. Reference data sensors that provide information about the UAV state must be installed. Considering that changes in the distance to obstacles correspond to changes in the UAV location, reference data sensors like IMUs and optical flow sensors are used to assess the accuracy of the main data and aid in selecting the best sensor. In the particular case of fixed obstacles, the aforementioned variances in distance ought to match. The weights are then calculated by applying a differential norm to compare all conceivable sensor combinations of main data and reference data. In each instant, the obstacle distance measurement corresponding to the sensor with the lowest weight is chosen, and the remaining measurements are discarded on the grounds that they are corrupted. Nonetheless, the sensor readings are fused in accordance with their weights if the computed weights have a low variation.

Considering a hypothetical system equipped with an IMU, a laser rangefinder and a RADAR, the

norms to compute the weight of the laser rangefinder would be

$$N_1 = |E_k - (L_k - L_{k-1})| \tag{3.8}$$

$$N_2 = |E_k - (L_k - D_{k-1})| \tag{3.9}$$

where E_k is the position variation of the UAV between the two measurements computed from the data given by the IMU, L_k and L_{k-1} are the measurements obtained from the laser rangefinder at instants k and $k-1$ and D_{k-1} is the measurement used in the previous iteration by the chosen sensor (or fused result). Once these norms are computed, an exponential moving average filter can be applied with a smoothing factor e, so that the history of these values can be accounted for. The first term of the filter is given by

$$W'_{L_k} = a_1 \times N_1 + a_2 \times N_2, \tag{3.10}$$

where a_1 and a_2 must be tuned to determine the influence of each source, and the filter itself is given by

$$W_{L_k} = e \times W'_{L_k} + (1 - e) \times W_{L_{k-1}}, \tag{3.11}$$

which corresponds to the weight for the laser rangefinder. The weight for the RADAR W_{R_k} is computed analogously with the measurement obtained from the RADAR (R_k) and, if the difference between both weights is less than 10%, the overall distance is computed by the weighted average

$$D_k = \frac{L_k \times W_{L_k} + R_k \times W_{R_k}}{W_{L_k} + W_{R_k}}. \tag{3.12}$$

3.6 Implementation in Simulation Tool

Upon combining the obstacle distances from the onboard sensors according to 3.5, another phase of data processing is initiated. This step is essentially a screening process for which obstacles should trigger an avoidance maneuver. A list of obstacles is strategically organized based on the urgency of detected collision possibilities, assuming the straight projection method, *i.e.*, the current state of the obstacle is projected into the future along a straight trajectory made at a constant velocity. This detection module is represented the flowchart in Fig. 3.3. As mentioned in 2.4.3, a safety radius is defined around the UAV and the shortest distance between the UAV and the obstacle is computed, whether it is a static or a dynamic obstacle. An evasive maneuver will only start if this distance is smaller than the defined safety radius. Consequently, this detection algorithm is able to optimize computational cost and conserves power during the avoidance stage, given the resource-intensive nature of the avoidance algorithms.

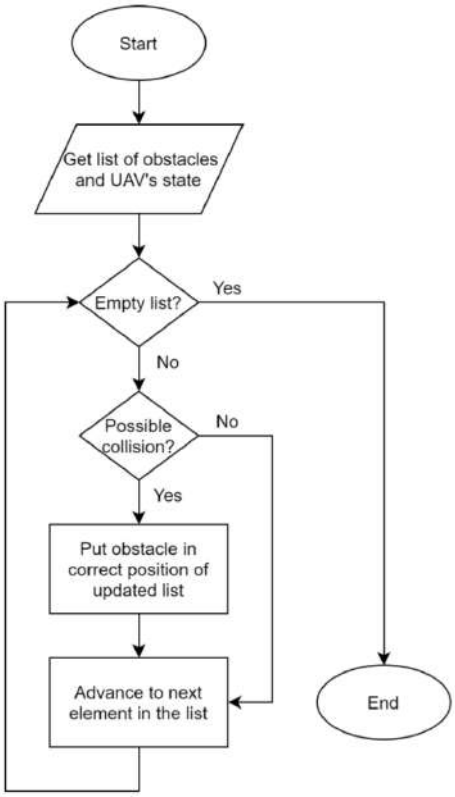


Figure 3.3: Obstacle screening flowchart [19].

Mathematically, the resulting collision detection method consists of computing the closest point of approach (CPA) between the UAV and the obstacle, assuming that both vehicles will maintain constant velocities and rectilinear paths. With these considerations, the motion of two vehicles, A and B, can be described as $A(t) = A_0 + \mathbf{v}_A t$ and $B(t) = B_0 + \mathbf{v}_B t$ and the distance between them at an instant t is given by $\|A(t) - B(t)\|$. By solving its derivative, the instant corresponding to the minimum distance between the two points (t_{CPA}) is given by

$$t_{CPA} = \frac{-(A_0 - B_0) \cdot (\mathbf{v}_A - \mathbf{v}_B)}{\|\mathbf{v}_A - \mathbf{v}_B\|^2}. \tag{3.13}$$

Knowing the value of t_{CPA}, the minimum distance between the two vehicles can be easily computed, as

$$d_{CPA} = \|A(t_{CPA}) - B(t_{CPA})\|. \tag{3.14}$$

If this distance is smaller than the safety radius, an evasive maneuver must be performed; otherwise the obstacle is not considered a threat to the UAV. Since the linear projection is not the most effective, these computations will have to be made regularly to take into account maneuvers performed by the UAV itself and by non-threatening obstacles that can become threatening after said maneuver. Furthermore, the installed sensors can detect new obstacles, and therefore a new risk evaluation must take place. In case of multiple collisions being detected, the obstacles are sorted according to their t_{CPA}, so that the obstacles associated with possible collisions that would happen first are avoided before the remaining

ones. The adopted algorithm to be implemented in MATLAB is represented in pseudo-code in Alg. 1.

Algorithm 1 Collision avoidance.

Require: UAV's position A, UAV's speed $\mathbf{v}_A$, obstacle's position B, obstacle's speed $\mathbf{v}_B$, obstacle safety radius R_s

Ensure: Flag indicating possible collision $flag$, t_{CPA}, Obstacle's position at CPA B_{CPA}

set $t_{CPA} = \frac{-(A_0 - B_0) \cdot (\mathbf{v}_A - \mathbf{v}_B)}{\|\mathbf{v}_A - \mathbf{v}_B\|^2}$

set $B_{CPA} = B + \mathbf{v}_B \cdot t_{CPA}$

set $A_{CPA} = A + \mathbf{v}_A \cdot t_{CPA}$

set $d_{CPA} = \|A_{CPA} - B_{CPA}\|$

if $t_{CPA} > 0$ and $d_{CPA} < R_s$ **then**

    set $flag$ = 1

else

    set $flag$ = 0

end if

Upon detecting a potential collision, an avoidance strategy is implemented to maintain a safe separation between the UAV and the identified obstacles. To accomplish this, iterative local paths are generated until there are no longer any obstacles representing potential collision risks and a final local path is exported to the flight controller.

It is important to note that global WPs are different from WPs belonging to the global path. Global WPs are considered obligatory that the UAV must reach to complete the ongoing flight phase, whereas WPs associated with the global path serve to connect two global waypoints. The latter define the trajectory but do not require mandatory traversal; the UAV has the flexibility to deviate from the global path if it is blocked by an obstacle.

In the event of an obstacle blocking the next global WP as a consequence of its evasion maneuver, a local path to reach the missed global WP is generated, thus ensuring the continuation of the current flight phase. However, if the obstructing obstacle remains static, the affected global WP is deemed unattainable, therefore being removed from the flight plan. This adjustment allows the UAV to progress with its mission by advancing to the subsequent global WP. These logical sequences form a collision avoidance module illustrated in Fig. 3.4.

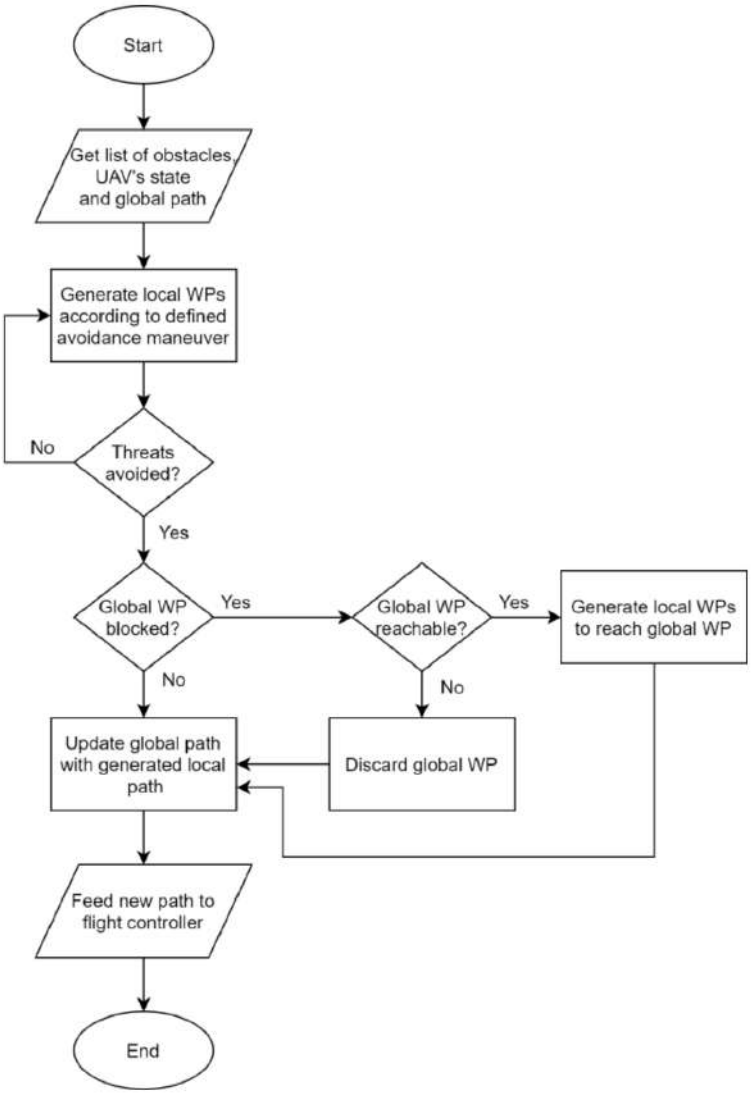


Figure 3.4: Collision avoidance flowchart [19].

To establish an avoidance algorithm, a path replanning strategy needs to be defined. UAVs must always give way to manned aircraft and, since the sensing is assumed to be non-cooperative, the UAV will always make an evasive maneuver to avoid a collision, while respecting the Rules of the Air [60]. Because this work presents a 2D simulation, climb and descent rules do not need to be taken into consideration. Thus, in case of an imminent collision, the avoidance strategy is the following:

- in level flight, if the intruder is in a head-on collision path or to the right of the UAV, the UAV turns right;
- in level flight, if the intruder is approaching from the left, the UAV turns left;
- in case of a static obstacle, the direction in which the obstacle is circled around is the one that corresponds to the smaller path to reach the goal.

The avoidance algorithm, employing the Potential Fields method (see Sec. 2.4.3), is represented in pseudo-code in Alg. 2. It includes two functions: P_{close} = find_closest(P, $path$), which finds the point

of the global path closest to the position of the UAV; and $\mathbf{v}$ =get_velocity($\mathbf{f}, \mathbf{v}$), which computes the direction of the force F and creates a velocity vector with this direction and the magnitude of V. The variable mov is computed previously and indicates the direction the UAV will take in order to avoid the obstacle according to their positions. Moreover, as stated in Sec. 2.4.3, the constants $S_{\max}$ and θ_{cut} need to be defined according to the size and speed of the UAV in order to obtain the most effective avoidance maneuvers.

Algorithm 2 Collision Avoidance Algorithm

Require: Position of UAV P, speed of UAV $\mathbf{v}$, position at CPA of obstacle P_{CPA}, radius of obstacle R_c, position of next global WP P_{next}, safety radius R_s, action radius R_a, weighing term α_{PF}, points of global path $path$, time step dt, variable indicating avoidance strategy mov

Ensure: Flag indicating if collision was avoided $flag$, next point of local path P_L

set $P_{close} =$ find_closest$(P, path)$
set $\mathbf{f}_{at} = \alpha_{PF} \frac{P_{\text{close}} - P}{\|P_{\text{close}} - P\|} + (1 - \alpha_{PF}) \frac{P_{\text{next}} - P_{\text{close}}}{\|P_{\text{next}} - P_{\text{close}}\|}$
set $\mathbf{d}_0 = P - P_{CPA}$
if $mov == 1$ **then**
  set $s_{dir} = \frac{\hat{k} \times \mathbf{d}_0}{\|\mathbf{d}_0\|}$
else
  set $s_{dir} = -\frac{\hat{k} \times \mathbf{d}_0}{\|\mathbf{d}_0\|}$
end if
if $\|\mathbf{d}_0\| \leq R_c$ **then**
  set $f_{rep} = \infty \frac{\cdot \mathbf{d}_0}{\|\mathbf{d}_0\|}$
else if $R_c < \|\mathbf{d}_0\| \leq R_s$ **then**
  set $S_{\max} \mathbf{s}_{dir}$
else if $R_s < \|\mathbf{d}_0\| \leq R_a$ **then**
  set $S_{\max} \frac{R_a - \|\mathbf{d}_0\|}{R_a - R_s} \mathbf{s}_{dir}$
else
  $f_{rep} = 0$
end if
set $\theta = \arccos\left(\frac{f_{at} \cdot d_0}{\|f_{at}\| \cdot \|d_0\|}\right)$
if $\theta > \theta_{cut}$ **then**
  set $f = f_{at}$
  set $flag = 1$
else
  set $f = f_{at} + f_{rep}$
  set $flag = 0$
end if
set $\mathbf{v} =$ get_velocity$(\mathbf{f}, \mathbf{v})$
set $P_L = P + \mathbf{v} \cdot dt$

Chapter 4

Optimal Sensing System

An optimization study was conducted to find the types of sensors and respective orientation that result in the best collision avoidance performance. To do so, a set of randomly generated collision scenarios with both stationary and moving obstacles were generated. The sensors modelled in Chap. 3 were tested for each of these scenarios, varying their orientation until optimal configurations were reached. The scenario generation and multi-sensor optimization algorithms were further developed based on [50]. The optimization results and the performance comparison of each solution can be found at the end this chapter, aiding in the selection of hardware to integrate the final collision avoidance system.

4.1 Scenarios Generation

In order to create scenarios that are suitable for this study, a scenario generation algorithm was created. Each scenario must specify the obstacle's initial position, velocity and radius. It also includes a pre-planned path and waypoints that the UAV must follow.

Figure 4.1 is based on the graphical representation of this algorithm present in [20], depicting the processes that lead to generating a scenario.

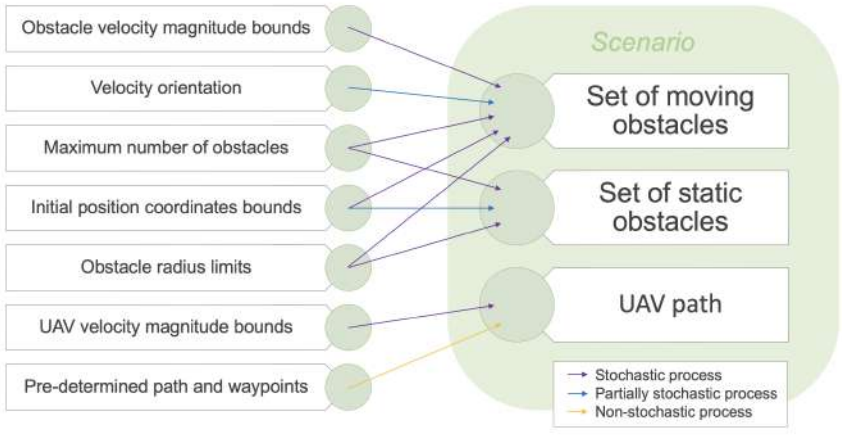


Figure 4.1: Scenario generation algorithm [20].

This algorithm aims at generating values for the aforementioned variables, fully characterizing the scenario. For this purpose, as shown in Fig. 4.1, different bounds are defined regarding the kinematic and dimensional properties of the obstacles and the UAV itself. Various stochastic and partially stochastic processes were then extracted from these intervals, creating random values for the different variables.

Partially stochastic processes have been used in two different cases: determining the velocity orientation of moving obstacles and setting the position of static obstacles. In the former, the goal is to ensure that deviation from the obstacles to the center of the graphical window is not predicted by initial conditions, *i.e.*, initially, the direction of the obstacle's velocity shall point to the centre of the window, rather than pointing outwards, increasing the possibility of collision. In the latter, the initial position of the static obstacle must not be within the safety radius around the waypoint, given that the UAV must pass through it.

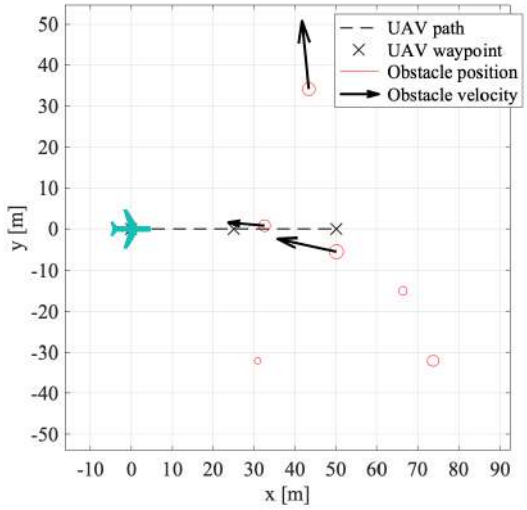


Figure 4.2: Randomly generated scenario.

An example of a resulting scenario is plotted in Fig. 4.2. This scenario generating function simply accepts the predetermined path and waypoints of the UAV as an input before combining them with a list of moving and static obstacles to produce a scenario. If the UAV does not go beyond any obstacle's safety radius throughout the whole simulation (without any sensors), the scenario will be discarded. Until there are n scenarios with an impending collision, this process is repeated.

4.2 Optimization Technique and Problem Formulation

To determine the optimal sensor configuration, different sensor sets were tested. The parameters that characterize each sensor model were obtained from their technical manuals or inferred from available data. This information regards the sensors presented in Sec. 2.3 and is summarized in Tab. 4.1. Since our simulations were restricted to the horizontal plane of motion, the vertical FOV is not relevant.

Table 4.1: Sensors quantitative comparison.

	Ultrasonic sensor	Laser rangefinder	LIDAR	RADAR
Manufacturer	MaxBotix	Lightware	Lightware	Ainstein
Model	MB1242	LW20/C	SF45/B	US-D1
Range (m)	6	100	45	50
Horizontal FOV (°)	-	0.3	variable	43
Accuracy (m)	0.1	0.1	0.1	0.04
Max. frequency (Hz)	7	388	5000	100

Forty collision-leading scenarios were randomly generated with obstacle parameters varying according to the limits set in Tab. 4.2.

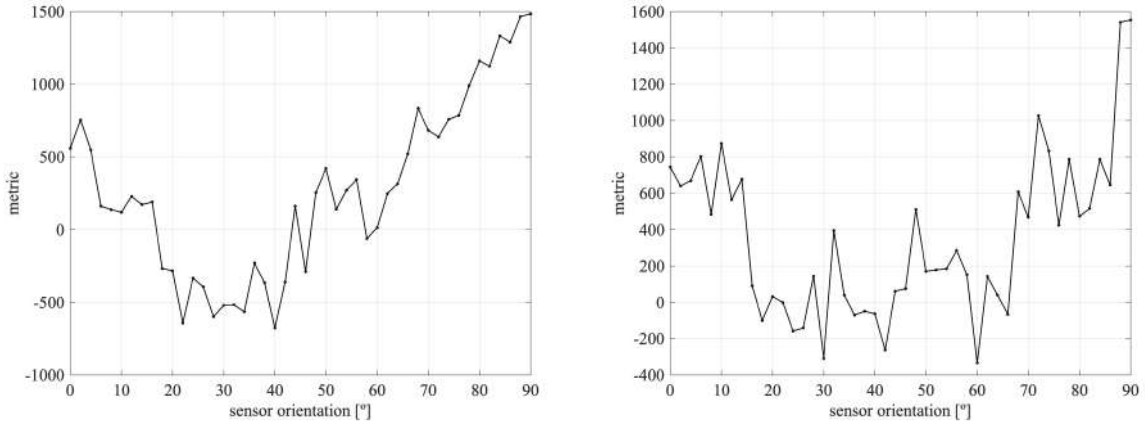
Table 4.2: Data for randomly generated imminent collision scenarios.

UAV speed	# fixed obst.	# moving obst.	obst.radius	obst.speed	obst.direction
$[8, 15]\mathrm{m/s}$	$\{0, 1, 2\}$	$\{0, 1, 2\}$	$[0.5, 2]\mathrm{m}$	$[5, 15]\mathrm{m/s}$	$[0, 90]^\circ$

In order to optimize the sensor orientation β, a function $f(\beta)$, to be minimized, was defined as

$$f(\beta) = \sum_j \sum_i \left(-d_{\min}(i) + \phi_1 \left| \max\left(R_s(i) - d_{\min}(i), 0\right) \right|^2 + \phi_2 \left| \max\left(R_c(i) - d_{\min}(i), 0\right) \right|^2 \right), \tag{4.1}$$

where the first term drives the evasion maneuver to maximize the minimum distance $d_{\min}$ between the UAV and the obstacle i, the second term represents the penalty when the minimum distance violates the safety radius R_s ($d_{\min} \leq R_s$), and the last term represents the penalty when the minimum distance violates the obstacle collision radius R_c ($d_{\min} \leq R_c$). The metric accumulates not only for every obstacle i in each scenario but also for all scenarios j. In order to penalize collision cases more than close-calls, the weights were set to $\phi_1 = 10$ and $\phi_2 = 50$.



(a) Pair of laser rangefinders.

(b) Pair of laser rangefinders and a RADAR.

Figure 4.3: S&A metric as function of laser rangefinder orientation.

Figure 4.3 shows the metric defined in Eq. (4.1) for two particular sensor solution cases: i) using a pair of laser rangefinders with 100 m range, symmetrically pointing forward with an angle β with respect

to the UAV longitudinal axis; and ii) adding a RADAR with 120 m range pointing in the direction of the UAV longitudinal axis.

In both cases, the metric proves to be noisy. Thus, the optimization technique selected to find the minimum of $f(\beta)$ was the Genetic Algorithm (GA) implemented in MATLAB. This gradient-free, population-based method, deals with a set of solutions that are updated simultaneously in each iteration. In practice, compared to other minimization algorithms, this reduces the likelihood of the result being a relative minima of the metric rather than the global optimum. This problem can be posed as

$$\begin{aligned} &\text{Minimize } f(\beta) \\ &\text{w.r.t. } \beta \\ &\text{subject to } \beta_{min} \leq \beta \leq \beta_{max} \end{aligned} \tag{4.2}$$

where β_{min} and β_{max} are the lower and upper bounds of β, respectively, to be defined for each particular case. Notice that β is a vector if multiple sensors are used.

Before performing any kind of simulation, optimization parameters need to be defined. Thus, the initial population was set to be created with a uniform distribution; the crossover function was set to create 80% of the population in each generation; because the variables are bounded, the mutation function randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation, where the chosen direction and step length satisfy the set bounds. The convergence criteria were set such that the global minimum was found in a timely but accurate manner: a function convergence of 10^{-3} was used with 10 stall generations, and a maximum of 50 generations prescribed. The population size was set to 30 individuals. These parameters were chosen following best practices [50]. The simulations were run on an 1,4 GHz Intel quad-core i5 with 8 GB 2133 MHz RAM.

4.3 Optimal Sensing Configurations

The following subsections are dedicated to detailing the proposed sensing architectures, further explaining each solution and the corresponding optimal result. In the end, the performance of the different sensor sets are summarized and compared.

4.3.1 Two Ultrasonic Sensors

For a set of two ultrasonic sensors, the orientation of each sensor was bounded between 0° and 90° from the longitudinal axis and the range was set to 6 m. To simplify the problem, the two sonars were considered to have a symmetrical orientation, resulting in just one design variable. A narrow beam pattern was adopted to better trace the obstacle locations.

The GA minimization terminated at 20 iterations, due to average change in the fitness value less than the specified tolerance, after performing 592 function evaluations for 39 hours and 40 minutes. It reached an optimal orientation of 36.5° (see Fig. 4.4).

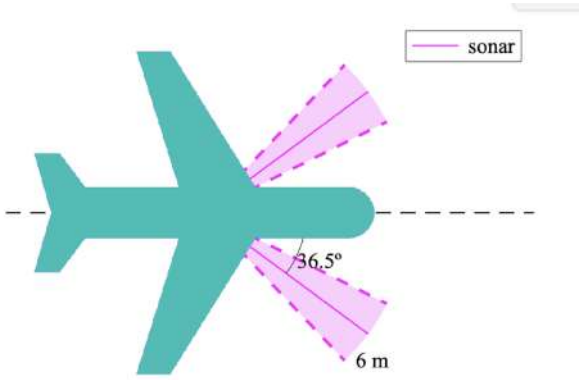


Figure 4.4: Optimal orientation for two ultrasonic sensors configuration.

Table 4.3: Performance comparison for different orientations of two laser rangefinders.

Orientation	Metric	Failure	Close call	Success rate
36.5 °	804.0	4/40	30/40	90.0%
0 °	1203.8	4/40	32/40	87.5%

A comparison of performance between the optimal orientation and a single sonar pointing forward was made. Table 4.3 illustrates both cases, considering a failure corresponds to a collision with an obstacle and a close call happens if the UAV breaches the safety radius of an obstacle. The optimal configuration resulted in 4 failures and close calls in 75% of the scenarios. The number of collisions is the same for both cases, however, there are 2 more close calls when the configuration only includes one sensor, thus, its success rate is lower.

Nonetheless, the results are not satisfactory for either cases, since the sensor's scanning pattern allows for the safety radius to be breached too many times. This was expected due to the short range of ultrasonic sensors, that makes it impossible for the UAV to detect the obstacle and replan its trajectory in a timely manner.

4.3.2 Two Laser Rangefinders

Analogous to the previous case, a set of two laser rangefinders with symmetrical orientation was considered. The orientation of each sensor was bounded between $0°$ and $90°$ from the longitudinal axis, in the horizontal plane. A sensing range of 100 m was adopted.

After 19 generations, the GA optimization algorithm finished, corresponding to 564 function evaluations and a computing time of approximately 6 hours. The optimal sensor orientation was $34.4°$, which corresponds well with one of the approximate minimum shown in the preliminary study in Fig. 4.3. The optimal two laser rangefinder sensor configuration is illustrated Fig. 4.5.

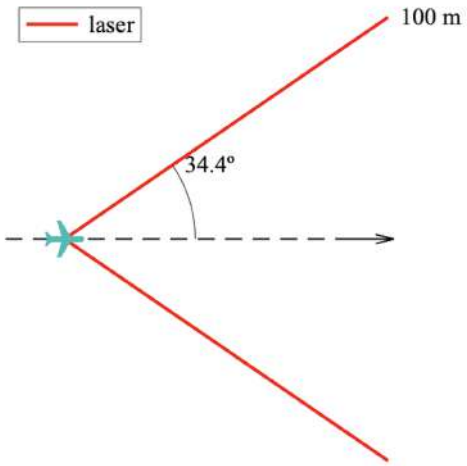


Figure 4.5: Optimal orientation for two laser rangefinder configuration.

Table 4.4: Performance comparison for different orientations of two laser rangefinders.

Orientation	Metric	Failure	Close call	Success rate
34.4 °	-414.0	1/40	23/40	97.5%
0 °	111.1	2/40	28/40	95.0%

The performance of this optimal configuration was compared to that of a single laser pointing forward, as seen in Tab. 4.4. Although the optimal configuration only fails once in 40 scenarios, the safety radius was breached in 23 of them. This result was expected, since a UAV equipped only with two laser rangefinders is not capable of properly tracking the moving obstacles when collisions are imminent.

When it is equipped with a single laser rangefinder, the sucess rate decreases to 95% and the close calls go up to 28, increasing the likelihood of collisions in a real-life scenario. This result was also anticipated, since less obstacles approaching from an angle can be tracked. The number of actual collisions in the simulation increased by one.

Compared to the previous case of ultrasonic sensors, these simulations demonstrated that laser rangefinders not only prevent more collisions but also more close calls. Overall, these sensors perform better under the given circumstances.

4.3.3 Two RADARs

Once again, the two RADAR sensors were considered to be symmetrical about the UAV longitudinal axis and the orientation spanned from 0° to 90°. Each RADAR had a range of 50 m, an accuracy of 0.04 m and a FOV of 43°.

After 11 generations, the optimizer finished 340 function evaluations. The optimal RADAR orientation was 9.2°, as illustrated in Fig.4.6.

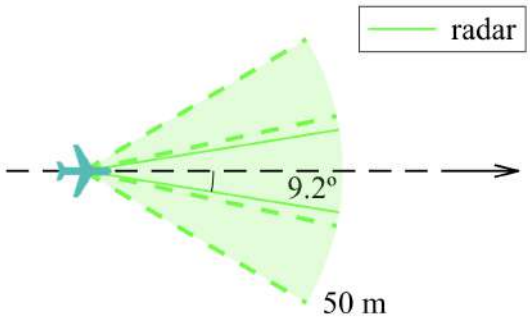


Figure 4.6: Optimal orientation for two RADAR configuration.

Another configuration worth studying would be a sensor orientation close to $21.5°$, which would yield the same result as if the UAV were equipped with a single RADAR with double FOV ($86°$). Table 4.5 includes the comparison between this configuration, the optimal orientation and a single RADAR pointing forward.

Table 4.5: Performance comparison for different orientations of two RADAR.

Orientation	Metric	Failure	Close call	Success rate
21.5 °	-1141.7	1/40	12/40	97.5%
9.2 °	-1171.0	1/40	12/40	97.5%
0 °	-312.3	4/40	13/40	90.0%

Regarding actual collisions, obstacles that approach the UAV from an angle are more likely to be detected by the optimal solution rather than by the single RADAR configuration. As can be seen in Tab. 4.5, the number of failures increase as the orientation decreases (for this particular case), which in turn makes the success rate decrease.

By overlapping the FOV of the two sensors, the accuracy is reduced through the data fusion algorithm. Thus, in this case, having a narrower FOV ($\beta = 9.2°$) and in turn, the juxtaposition of both RADARs proved to be almost as effective as the double FOV configuration ($\beta = 21.5°$).

These simulations showed that the reduced accuracy of the RADAR proves to be impactful on the precision of obstacle tracking compared to that of the laser sensors. Despite having a broader FOV and resulting in less close calls, the RADAR solution led to just as many collisions, which means that the two laser rangefinder configuration remains as promising (same success rate). It is reasonable to say that while RADAR FOV is more crucial for detecting obstacles, the sensor's accuracy is the most significant factor for effective collision avoidance.

4.3.4 Two LIDARs

Each LIDAR was modelled with a range of 45 m, an accuracy of 0.1 m and a variable FOV. According to hardware specifications (see Tab.4.1), this FOV can range from $20°$ to $320°$, thus, a FOV of $180°$ was

chosen. This value ensures a reasonable trade-off between timely scanning frequency and a broad scope.

However, this makes optimization redundant, as illustrated in Fig. 4.7. This happens due to the nature of the scenario generation algorithm used: because the obstacles are spawned inside the limits of the scenario, it is worthless to track the area behind the UAV in the initial instant. Furthermore, from this instant on, if an obstacle were positioned behind the UAV, it would have already been tracked before due to the wide FOV and long range of the LIDAR. The overlapping of the FOV in the case of a two LIDAR solution does not prove to be advantageous either. Note that this is only verified for a FOV of $180°$. If the FOV were smaller, it would be convenient to optimize the sensor orientation.

In this particular case, it is fair to state that the most beneficial solution would be to use a single LIDAR pointing forward, since it decreases hardware cost. This configuration is illustrated in Fig. 4.8.

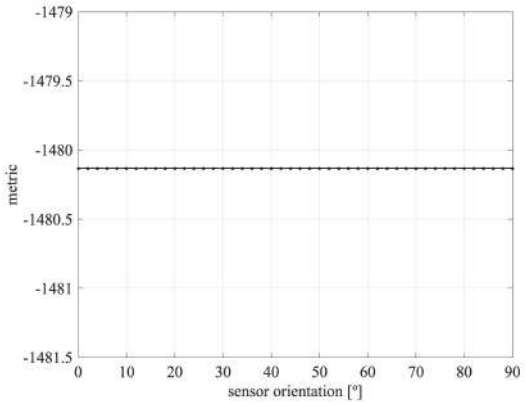


Figure 4.7: S&A metric as function of sensor orientation for a set of two LIDAR.

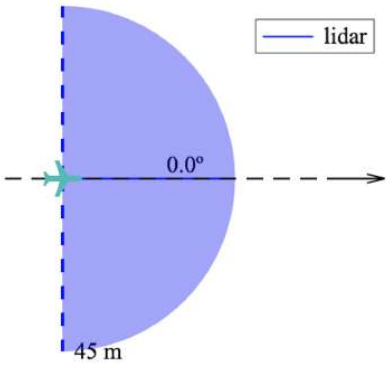


Figure 4.8: Single LIDAR configuration.

Table 4.6: Performance comparison for different orientations of two LIDAR.

Orientation	Metric	Failure	Close call	Success rate
0.5 °	-1480.1	0/40	9/40	100.0%
0 °	-1480.1	0/40	9/40	100.0%

Table 4.6 includes the performance comparison for different orientations of two LIDAR. As mentioned above, the success rate is the same since the optimal solution found ($\beta = 0.5°$) corresponds to an approximation of the exact optimal solution ($\beta = 0$). Compared to the previous types of sensors studied, the LIDAR performs better overall. The wide FOV reduces the chances of close calls and eliminates the possibility of failure.

4.3.5 Performance Comparison of Sensor Sets

Other solutions that involved three sensors were optimized, for example, including two laser rangefinders symmetrical about the UAV longitudinal axis, whose orientations were bonded between $0°$ and $70°$;

and one fixed RADAR pointing forward. This configuration was also replicated with two lasers and one LIDAR, two RADARs and one laser, and two RADARs and one LIDAR. The performance of the optimal version of these sets of sensors is summarized in Tab. 4.7, as well as the results from the solutions with only one type of sensor. Optimizations with different sets of sensors were performed but left out of this table in order to avoid redundancy of results.

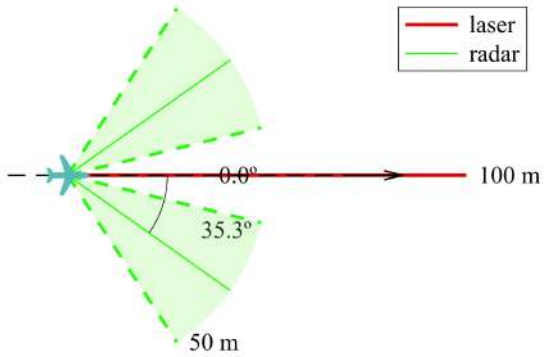
Table 4.7: Comparison of the optimal performance for the different sensor sets studied.

Sensors	Metric	Failure	Close call	Success rate
2 SONARs @ 36.5 °	804.0	4/40	30/40	90.0%
2 lasers @ 34.4 °	-414.0	1/40	23/40	97.5%
2 lasers @ 63.4 ° + 1 RADAR @ 0 °	-1240.4	0/40	11/40	100.0%
2 lasers @ 10.0 ° + 1 LIDAR @ 0 °	-1606.4	0/40	8/40	100.0%
2 RADARs @ 9.2 °	-1171.0	1/40	12/40	97.5%
2 RADARs @ 21.5 °	-1141.7	1/40	12/40	97.5%
2 RADARs @ 35.3 ° + 1 laser @ 0 °	-1480.1	0/40	9/40	100.0%
2 RADARs @ 28.1 ° + 1 LIDAR @ 0 °	-1574.3	0/40	9/40	100.0%
1 LIDAR @ 0 °	-1480.1	0/40	9/40	100.0%

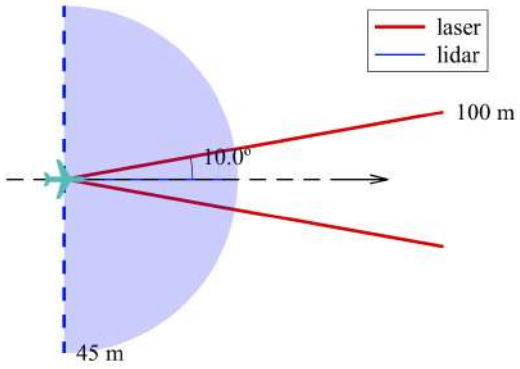
For the set of scenarios tested, the RADAR performed better than the laser rangefinder, which in turn performed better than the ultrasonic sensor if only one sensor type is to be used. Nonetheless, this is tightly dependent on the sensor characteristics, such as range, FOV and accuracy. Furthermore, a single LIDAR was enough to outperform all other types of sensor.

As expected, all the solutions that present a 100% success rate include either a RADAR or a LIDAR in their configuration. If the LIDAR is kept out, it is the two RADAR and one laser rangefinder solution that produced the least collisions and led to the least close calls. From these findings, together with the 0° FOV of a laser rangefinder, it is expected that increasing even more the number of sensors would lead to even better performance, thought at a higher hardware cost.

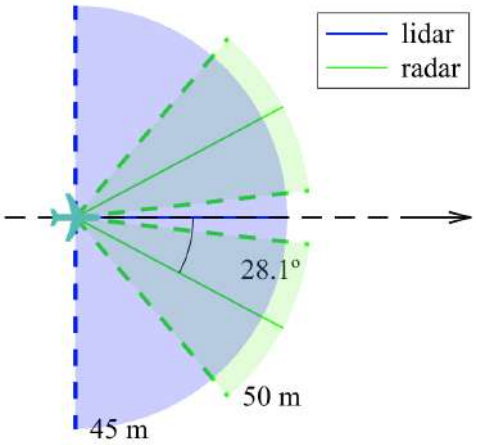
Comparing the solutions that include a LIDAR, it is proved that it is not significantly advantageous to pair it with other types of sensors, since it already performs distinctively well on its own. Regardless, the two laser and the two RADAR solution are beneficial due to reducing the likelihood of close calls. Despite the LIDAR having a wide FOV that is not increased by either configuration, the chances of breaching the safety radius decrease because the other sensors provide additional detection capacity. *I.e.,* since the LIDAR sweeps the designated area at a certain frequency, there are time instants when a fraction of the area within the LIDAR FOV is 'unsupervised'. Therefore, it is useful to have another set of sensors that track obstacles approaching from that specific area.



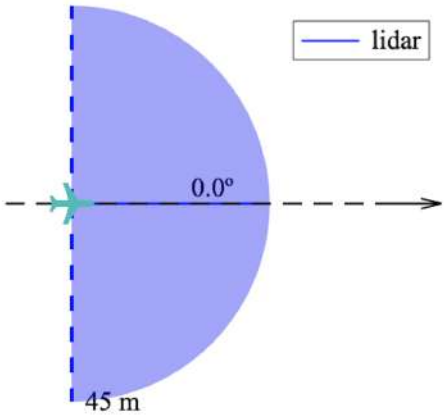
(a) Optimal orientation for two RADARs and one laser rangefinder configuration.



(b) Optimal orientation for two lasers and one LIDAR configuration.



(c) Optimal orientation for two RADARs and one LIDAR configuration.



(d) Single LIDAR configuration.

Figure 4.9: Sensor configuration to be tested.

To summarize, the optimized configuration had a very similar performance in four different cases (reflected in the *Metric* column), being the most promising one composed of one LIDAR pointing forward, complemented by two laser rangefinders pointing at 10° sideways. This outcome aids in selecting the hardware to be implemented in an optimal final solution. It would also be valuable to conduct experiences with a single LIDAR, in order to verify the efficiency claimed in the computed results. These configurations are illustrated in Fig. 4.9.

Chapter 5

Hardware Implementation

This chapter is dedicated to the description of the hardware that incorporates the sensing system. Once the hardware is known and available, the electrical layout is designed to enable the assembly of the final system.

5.1 Sensor Hardware

After briefly comparing the available sensors in Sec. 2.3, and having built a mathematical model that attests to their functionality in Chap. 3, it is finally possible to list the specifications of the chosen models. Table 5.1 was based on the sensors' respective datasheets.

Table 5.1: Sensor hardware specifications.

	Ultrasonic sensor MB1242 [61]	Laser LW20/C [62]	LIDAR SF45/B [63]	RADAR US-D1 [64]
Range (m)	7	100	45	50
Horizontal FOV (°)	0	0.3	variable	43
Resolution (cm)	1	1	1	–
Accuracy (m)	0.1	0.1	0.1	0.04
Update rate (Hz)	7	388	5000	100
Power supply voltage (V)	3-5.5	4.5-5.5	4.5-5.5	5-5.5
Power supply current (mA)	4.4	100	300	400
Outputs and interfaces	Serial and I2C	Serial and I2C	Serial and I2C, Micro USB	UART, CAN
Dimensions (mm)	22x19x15	30x20x43	51x48x44	108x79x20
Weight (g)	5.9	20	59	110
MSRP (€)	40	300	450	600

Regarding outputs and interfaces, it is important to note that Inter-Integrated Circuit (I2C) is a synchronous serial communication protocol that allows multiple devices to be connected to the same bus,

using only two bidirectional lines known as the Serial Data (SDA) line and the Serial Clock (SCL) line. Devices on the I2C bus can act as senders (masters) or receivers (slaves). The I2C interface is used for short-distance communication between devices, and it is commonly used in sensors, among other applications. It offers the advantage of addressing multiple devices with an I2C bus splitter (see Fig. 5.3), providing flexibility in system design [65]. The UART (Universal Asynchronous Receiver-Transmitter) interface is also a serial communication protocol, but it is asynchronous, *i.e.*, there is no clock signal to synchronize the output bits from the transmitting device going to the receiving end. It uses separate lines for transmitting and receiving data: Transmitter (TX) and Receiver (RX) [66]. UART is used for longer-distance communication, and it is commonly used in applications such as GPS receivers and wireless modems. The pins for I2C and UART interfaces depend on the specific sensor model and are shown in detail in Sec. 5.3.

The size of a sensor is a crucial factor to consider when designing a S&A system that employs it, since it may be counterproductive to add detection capability at the cost of vehicle maneuverability. However, this is likely not the case for the particular set of sensors in this study. In Tab. 5.1, the size of the sensor increases from left to right, with the MB1242 sonar being the smallest and lightest. The following LW20/C laser is slightly larger and heavier, but the adjacent SF45/B LIDAR doubles its size. Finally, the US-D1 RADAR takes up the most volume and weight, but is still sufficiently compact for this application. Table 5.1 is also correctly sorted by power consumption, from left to right. This specification is particularly important since the projected system is battery-powered and power shortages might arise from the combination of multiple sensors.

Regarding cost of acquisition, the MB1242 sonar is the cheapest sensor in Tab. 5.1, with a Manufacturer's Suggested Retail Price (MSRP) of around 40€. The LW20/C one-dimensional laser rangefinder is more expensive, due to its extended range, at at MSRP of 300€. Expectedly, the SF45/B LIDAR is even more costly, at 450€, since it is produced by the same manufacturer as the laser and employs the same technology, but at a larger scale (2D with variable FOV). Lastly, the US-D1 RADAR is the most expensive, with a MSRP of around 600€. RADAR technology is usually pricier than other distance sensing technologies due to its hardware request, which ultimately is also reflected on its size. The data that the sensor collects has to undergo extensive processing to provide accurate distance measurements, as hinted by the RADAR model in Sec. 3.4. Note that RADAR technology is relatively new compared to sonar and laser. However, as the technology matures and becomes more widely adopted, the manufacturing and development costs of RADAR sensors are expected to decrease.

5.2 Flight Controller

A flight controller is required to collect data from the chosen sensors. The Pixhawk 2.1 or Cube Black (see Fig. 5.1), was the chosen controller in [20], thus will also be used in the current work. It consists primarily of three parts:

- **Pixhawk FMU (Flight Management Unit) Main Board**, including a 32 bit microcontroller, 256kB of RAM, 2MB of flash, an integrated accelerometer/gyro and an altimeter;

- **Vibration Damped IMU Board**, including extra sensors such as accelerometer, magnetometer, gyroscope and altimeter, which will reject vibrations due to being mounted on a vibration damped board. This additional, vibration-free data will result in redundant measurements, improving the overall reliability of the system;
- **Input/ Output (I/O) ports** (see Fig. 5.2), including:
  - TELEM1 and TELEM2 ports used for telemetry communication;
  - GPS1 port used for GPS communication;
  - POWER1 and POWER2 ports to provide the necessary voltage for the flight controller and connected devices;
  - Other ports including I2C and Analog-to-Digital conversion (ADC), which can be used to connect to sensors and other peripherals. Refer to the complete list of the pin assignments for each port in Tabs. 5.2 - 5.10.

Figure 5.1: Pixhawk Cube Black [67].

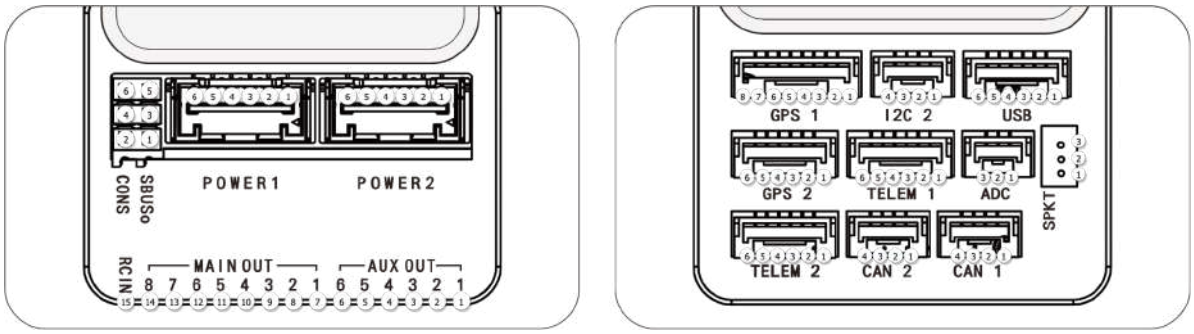


Figure 5.2: Pixhawk Cube Black port interface and pin label [68].

Table 5.2: TELEM1, TELEM2 ports.

Pin	Signal	Volt
1	VCC	+5 V
2	TX (OUT)	+3.3V
3	RX (IN)	+3.3V
4	CTS	+3.3V
5	RTS	+3.3V
6	GND	GND

Table 5.3: GPS1 port.

Pin	Signal	Volt
1	VCC	+5 V
2	TX (OUT)	+3.3V
3	RX (IN)	+3.3V
4	SCL I2C1	+3.3V
5	SDA I2C1	+3.3V
6	Button	GND
7	button LED	GND
8	GND	GND

Table 5.4: GPS2 port.

Pin	Signal	Volt
1	VCC	+5 V
2	TX (OUT)	+3.3V
3	RX (IN)	+3.3V
4	SCL I2C2	+3.3V
5	SDA I2C2	+3.3V
6	GND	GND

Table 5.5: ADC port.

Pin	Signal	Volt
1	VCC	+5 V
2	ADC IN	
3	GND	GND

Table 5.6: I2C2 port.

Pin	Signal	Volt
1	VCC	+5 V
2	SCL	+3.3 V (pullups)
3	SDA	+3.3 V (pullups)
4	GND	GND

Table 5.7: CAN1&2 ports.

Pin	Signal	Volt
1	VCC	+5 V
2	CAN_H	+12 V
3	CAN_L	+12 V
4	GND	GND

Table 5.8: POWER1 port.

Pin	Signal	Volt
1	VCC	+5 V
2	VCC	+5 V
3	CURRENT	up to +3.3 V, pin 3
4	VOLTAGE	up to +3.3 V, pin 2
5	GND	GND
6	GND	GND

Table 5.9: POWER2 port.

Pin	Signal	Volt
1 (red)	VCC	+5 V
2 (red)	VCC	+5 V
3 (blk)	CURRENT	up to +3.3 V, pin 14
4 (blk)	VOLTAGE	up to +3.3 V, pin 13
5 (blk)	GND	GND
6 (blk)	GND	GND

Table 5.10: USB port.

Pin	Signal	Volt
1	VCC	+5 V
2	D_plus	+3.3 V
3	D_minus	+3.3 V
4	GND	GND
5	BUZZER	battery voltage
6	Boot/Error LED	

5.3 Electrical Layout

The electrical layout can be designed once the flight controller and sensors have been chosen. It is possible to connect all the components as shown in Fig. 5.4 using the connections and supplementary devices (GPS and Power module) that are included within the Cube Black package.

The sensors are connected to the I2C 2 port either individually or collectively using an I2C bus splitter module shown in Fig. 5.3, while the GPS is connected to the GPS1 port. It is necessary to employ a power module so as to give the flight controller a regulated power source and power the electronic speed controller (ESC) at the same time.

Figure 5.3: I2C bus splitter [67].

A 6-wire cable that has two 5V connectors, two ground connections, and a connector for the battery's voltage and current (V and I, respectively) is used to power the flight controller. With the aim of supplying the flight controller information about the battery status, these final two connections are crucial.

The ESC also draws power from a battery and operates the motor using a pulse-width modulation (PWM) signal from one of the PWM I/O entries. It is relevant to take into account that although there is only one motor connected to the ESC in Fig. 5.4, the ESC must receive a separate PWM signal for each additional motor that is required.

A pulse-position modulation (PPM) Sum Receiver is also present, and it needs to be connected to an RX IN input. This component converts the PWM signals from the radio receiver into a single PPM signal that the flight controller can process.

Lastly, the telemetry module communicates through radio waves with a second telemetry module that is linked to a ground station. This allows real-time data to be exchanged and orders to be sent to the vehicle.

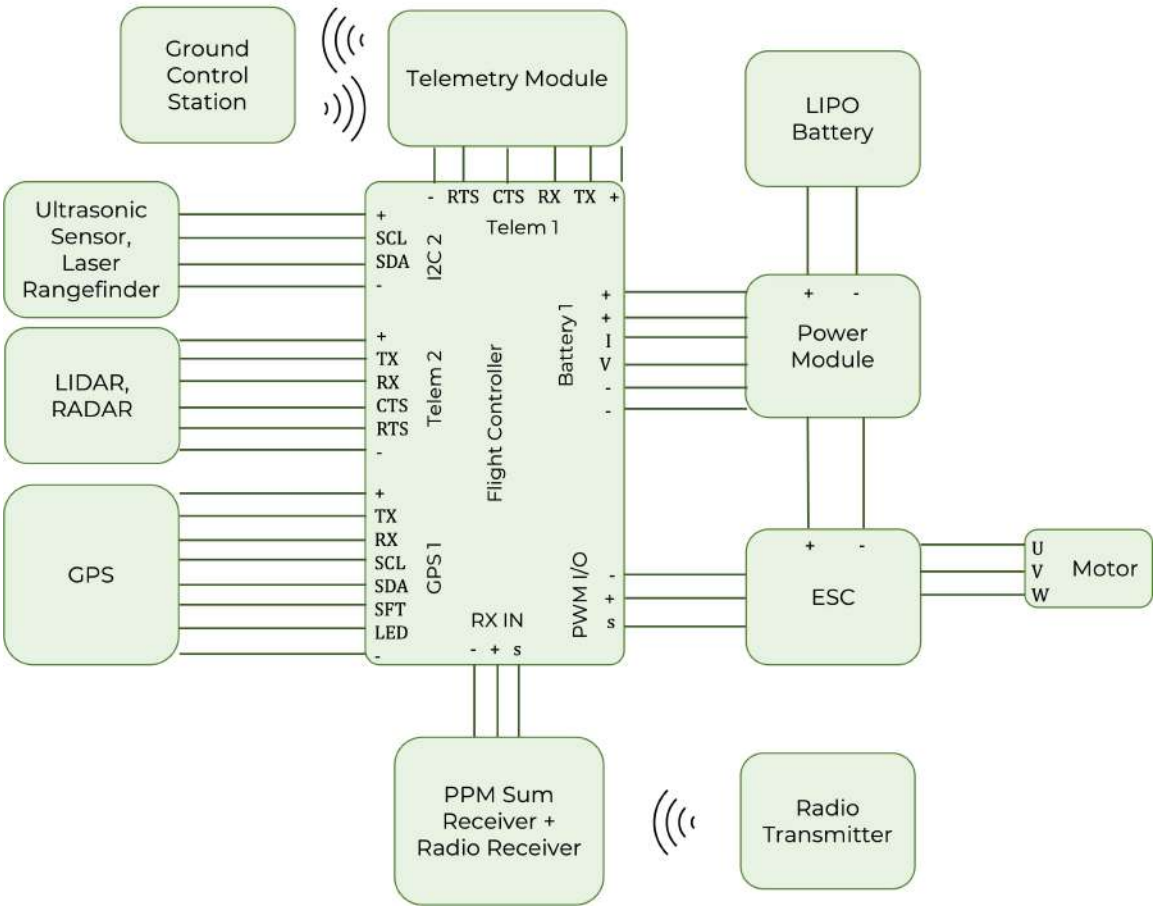


Figure 5.4: Electrical diagram [20].

Each distance sensor requires specific connections. Firstly, the MB1242 ultrasonic sensor by MaxBotix [57] must be connected to the Pixhawk I2C 2 port using a GH cable according to the diagram in Fig. 5.5. A similar cable can also be used to attach the LW20/C laser rangefinder by LightWare [62], following the required I2C and serial connections stated in Tab. 5.11.

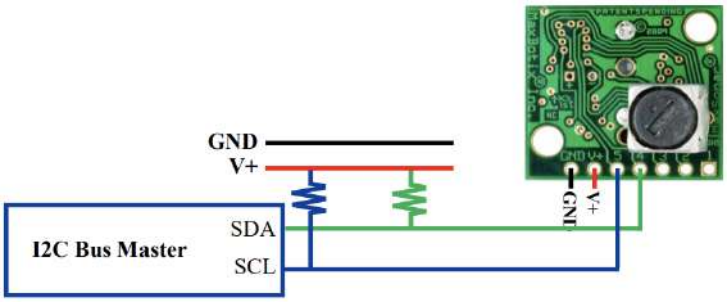


Figure 5.5: MB1242 wiring diagram.

Table 5.11: LW20/C connections.

Cable colour	Serial connection	I2C connection
red	VCC	VCC
black	GND	GND
yellow	TXD	SDA
white	RXD	SCL

The SF45/B LIDAR by LightWare [63] can be connected to the flight controller's TELEM2 port (TX and RX pins) using a DF13 header. The red and black wires (VCC and GND) were connected with an external power supply and the remaining three wires (blue, white and green) were left unconnected. Figure 5.6 ilustrates the eletrical wiring diagram for this configuration.

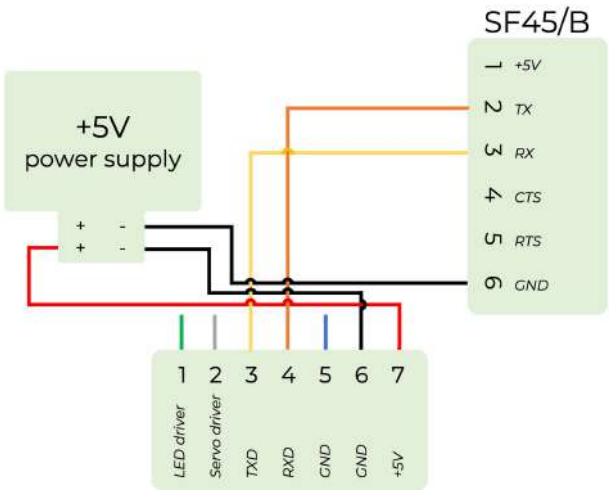


Figure 5.6: Eletrical wiring diagram for SF45/B.

The US-D1 RADAR by Ainstein [64] can also be connected to the TELEM2 port according to Tab.5.12.

Table 5.12: US-D1 connections.

Cable colour	UART	TELEM2	pin #
red	VCC	VCC_5V	1
green	TXD	MCU_RX	3
white	RXD	MCU_TX	2
black	GND	GND	6

The final assembly of the proposed system is depicted in Fig. 5.7. In this particular case, the LIDAR is connected to the TELEM2 port.

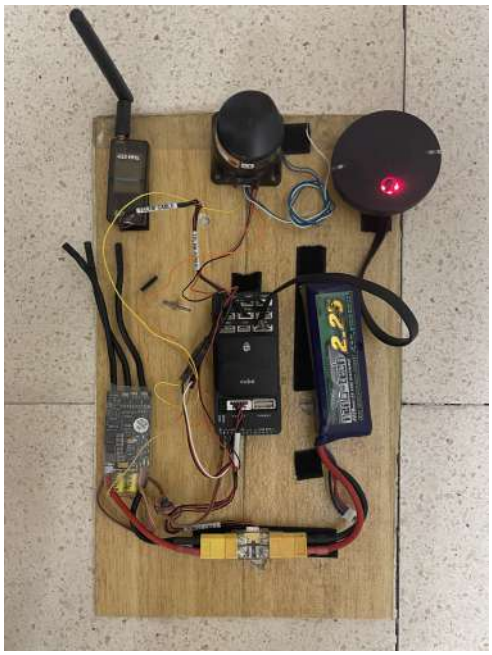


Figure 5.7: Assembly of the proposed layout (LIDAR-based obstacle detection system).

Chapter 6

Software Implementation

Some basic concepts are needed in order to build and fly an unmanned vehicle using PX4. PX4 is a core part of a broader drone platform that includes the QGroundControl ground station, the Pixhawk hardware mentioned in Chap. 5, and MAVSDK for integration with companion computers, cameras and other hardware using the MAVLink protocol.

This section provides a basic introduction to the flight controller (PX4) and Ground Control Station (QGroundControl), including the steps that must be taken to build and adapt this software to the current work.

6.1 Flight Controller

PX4 is a powerful open source autopilot flight stack that can be built on a console or in an Integrated Development Environment (IDE), for both simulated and hardware targets. VSCode [69] is the officially supported (and recommended) IDE for PX4 development. It is easy to set up and can be used to compile PX4, although it is not required.

Analogous to all software products, PX4 goes through a software development lifecycle that consists of three main stages: alpha, beta and stable release. Alpha means the product is actively in development after the previous software release and being tested internally. This version is not released in public repositories and typically includes new features and minor bug fixes. Beta is the software development phase that follows, when the product is feature-complete but still being tested internally and by users. Software in the beta phase will generally have many more bugs than completed software and speed or performance issues, and may still cause crashes or data loss. The focus of beta testing is reducing impacts on users, thus incorporating usability testing. On the other hand, it allows users to try newer features while helping the developers flight test new code. Finally, the stable release has passed all stages of verification and testing. The remaining bugs are considered acceptable.

Generally, the most recent stable released version of PX4 ought to be used, in order to benefit from bug fixes and get updated features. As such, this is the version that is installed by default and integrated into QGroundControl. However, the current stable release (v1.13.3) does not include the driver for the

LIDAR used in this work. Consequently, it was necessary to switch to a more recent release (v1.14) that includes it but is still in beta testing.

The PX4 source code is stored on Github a repository called PX4/PX4-Autopilot. To get release 1.14 onto a computer, the following commands must be entered into a terminal:

```
git clone https://github.com/PX4/PX4-Autopilot.git --recursive
git checkout release/1.14
```

To build for Pixhawk-based boards, it is necessary to navigate into the PX4-Autopilot directory and then call make with the build target for the board in use [70]. For Hex Cube Black, the corresponding board is *px4_fmu-v3_default*. As aforementioned, this release is being used to make the integration of the LightWare SF45 LIDAR possible. With this purpose, it is required to run each of the following instructions:

```
cd PX4-Autopilot
make px4_fmu-v3_default boardconfig
```

If there are unresolved conflicts in Git submodules, this command might start a loop of errors. In that case the following steps will ensure that the submodules are in the correct state and compatible with the current branch:

```
git submodule sync --recursive
git submodule update --init --recursive
```

Once the submodules are updated, it should be possible to run the 'boardconfig' command again.

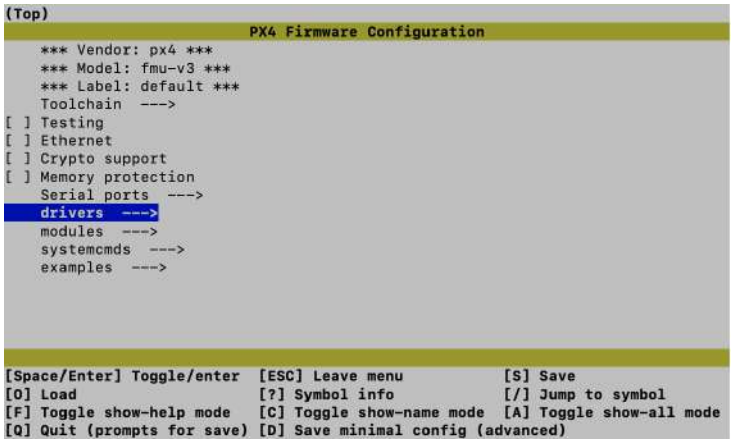


Figure 6.1: PX4 Firmware Configuration interface.

The user will then be prompted to update the firmware configuration in another interface (Fig. 6.1), and must select *drivers – Distance sensors*, and enable the *lightware_sf45_serial* and the *ulanding_radar* drivers. Lastly, it is necessary to save these alterations (by pressing the 'S' key). Thus far, these were the only sensors that required this procedure, as further explained in Chap. 7. However, if any other driver were yet to be enabled, this interface would also be useful [71].

Any changes made must be compiled into a new build, as follows:

```
make px4_fmu-v3_default
```

Since a beta release is being used, some common errors derive from missing files or other alterations that may cause the build process to fail. If this is the case, usually it can be fixed by making sure the latest version of the PX4-Autopilot repository is being used. This translates to running the following command from the root of PX4-Autopilot workspace:

```
1    git pull origin release/1.14
```

If the issue persists, it is still possible to clean the build artifacts and rebuild, as such:

```
1    make clean
2    make px4_fmu-v3_default
```

Once these steps are completed, the PX4 Autopilot firmware will be compiled, generating an executable file that can be uploaded onto the flight controller. This file can be found in the build directory within the PX4 Autopilot project directory. The name of the subdirectory and executable file depends on the specific configuration and build settings but typically corresponds to the name of the target board.

Among PX4's flexible set of tools to create tailored solutions for vehicle applications, Micro Object Request Broker (uORB) is the core of all the communications between the vehicle internal components. It is an asynchronous publish/subscribe messaging Application Programming Interface (API) used for inter-thread communication. More than 100 built-in topics are listed in the directory Firmware/msg on the development computer where PX4 firmware is downloaded. Despite having a large number of built-in topics, it might be necessary to add more. In this particular study, the topic 'DistanceSensor.msg' needs to be replicated as many times as there are additional sensors being used simultaneously. Each time, it is necessary to make new .msg files in the msg/ directory and add the file names to the msg/CMakeLists.txt list in order to add extra topics. This is used to automatically generate the necessary C/C++ code as the code is being created [72].

However, the logger will not log each topic in the system by default, in order to conserve memory and bandwidth. As a result, before it is possible to log a new uORB topic, it must be registered. To do so, it is necessary to append it to the list of topics using the function 'add_default_topics()' that can be found in src/modules/logger/logged_topics.cpp [73].

One of the goals of this work is to make use of the 3D scanning capabilities of the LIDAR, hence having installed its driver onto the firmware. It is necessary to know the angle at which the sensor is scanning to be able to determine the direction of the approaching obstacle. The aforementioned 'Distance-Sensor.msg' topic is responsible for publishing the sensor data to be displayed on the ground control interface. However, this topic is projected for simple rangefinders and therefore does not automatically publish the scan angle. Nonetheless, the scan angle is being extracted from LIDAR measurements once the driver is enabled. The function 'sf45_process_replies' in *PX4-Autopilot/src/drivers/distance_sensor/ /lightware_sf45_serial/lightware_sf45_serial.cpp* is responsible for this yaw angle measurement processing. One way of leveraging the processed value is to directly have it printed on the log by calling 'PX4_INFO' (printf equivalent function for PX4 shell) inside the 'collect' function (line 23). The relevant excerpt of PX4 code with this change can be found in App. A. Upon compiling it onto the build, it is possible to use MATLAB for treatment of the experimental data.

Another possibility is to feed the distance sensor topic with the yaw measurement regardless of it not being fit. With this in mind, a function 's_update' can be created in *PX4-Autopilot/src/lib/drivers/ /rangefinder/PX4Rangefinder.cpp* to replicate the existing 'update' function, but reporting yaw angle values instead. However, this will result in the angle values being registered unpredictably and under the wrong variables. Nonetheless, the respective code is also included in App. A since it allows for a better understanding of the overall data acquisition and communication system. Note that only the changes applied to '.cpp' files are present in App. A, but these must be in agreement with the headers in the corresponding '.hpp' files.

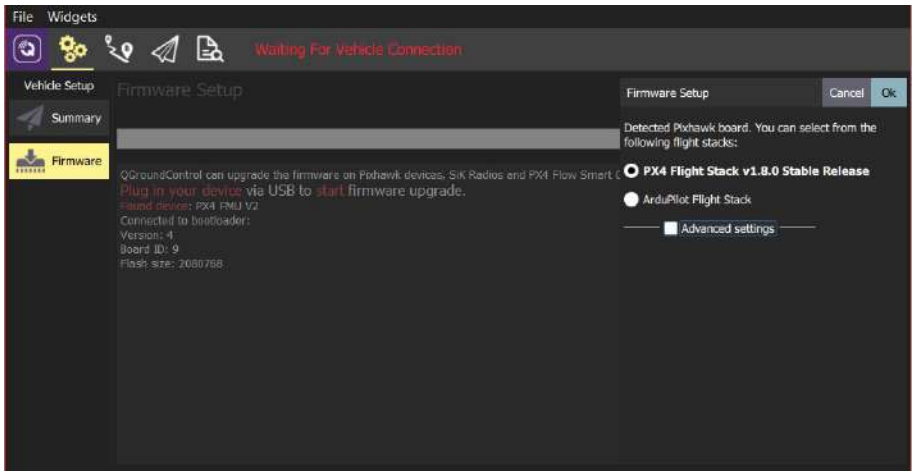
6.2 Ground Control

A ground control station functions almost as a virtual cockpit, serving as an interface between a flight controller and a human operator. Typically, a software program running on a computer is connected to the flight controller through wireless telemetry. This tool enables the human operator to communicate with the aircraft while it is in flight, allowing the acquisition of relevant data such as position, velocity, acceleration, or any other sensor data. It can be installed simply by running the executable file available in the QGroundControl user manual [74] (which includes versions for Windows, Mac OS X, Ubuntu Linux, Android and iOS).

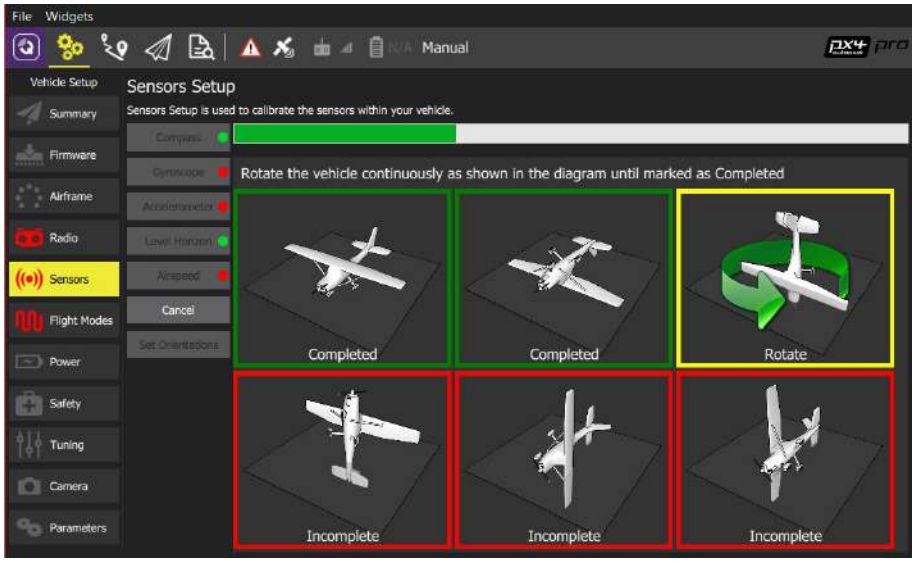
As shown in Fig. 6.2(a), it is possible to install the PX4 firmware into the flight controller. The flight controller needs to be USB-connected to the computer in order to accomplish that. The vehicle setup tool must be chosen in the QGroundControl application before the user can choose which firmware he wants to transfer to his controller in the Firmware Setup option. Because this project requires building and adapting the software, and once those changes are done (see Sec. 6.1), it will be necessary to opt for the Advanced settings and upload the altered PX4 file from the cloned folder.

The user is then prompted by QGroundControl to choose which airframe matches his vehicle (there are more than 30 options supported by PX4). The calibration is not complete until the user configures the controller's built-in sensors, radio receiver, flying modes, power, and motors. The flight controller's mounted gyroscope's calibration is shown in Fig. 6.2(b).

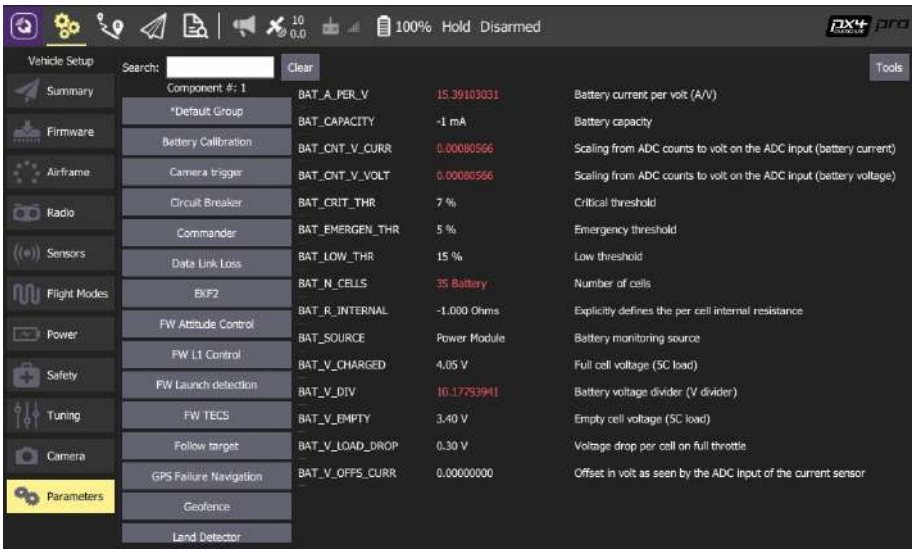
Moreover, it is necessary to emphasize that QGroundControl offers an easy approach to interact with PX4 firmware by adjusting various parameters, which can be done in the parameters tab inside the vehicle configuration option as shown in Fig. 6.2(c). These parameters will be important to enable each sensors in Chap. 7. Additionally, it is necessary to activate the SD logging mode by changing the 'SDLOG_MODE' parameter to "from boot until shutdown". By doing so, the SD card in use will collect data accordingly. Therefore, it is also recommended to make use of the 'logger' command on the MAVLink Console (inside Analyze Tools tab), in order to create shorter logs when performing different experiments. This prevents the time consuming download of unnecessarily heavy files.



(a) Firmware upload.



(b) Sensors calibration.



(c) Changing parameters.

Figure 6.2: QGroundControl environment.

Chapter 7

Sensor Experiments

To validate the capabilities of the S&A system, several experiments had to be performed. More specifically, the optimized configuration of the sensors that integrate it according to Chap. 4 had to be empirically tested. Therefore, all sensors were individually tested before the complete system. Due to the risk associated with flight testing, initial experiments were based on ground tests using a simple rover.

7.1 Bench Tests

Although each of the sensors employed has an available data sheet, it is good practice to perform several individual experiments to determine the actual range and FOV restrictions for the chosen devices. Ultimately, the purpose of these studies was to ascertain the sensors' detection rates and the accuracy of their measurements.

7.1.1 Ultrasonic Sensor

The MB1242 ultrasonic sensor by MaxBotix [57] should be enabled within the QGroundControl environment. To do this, the user must access the vehicle setup section and, within the parameters tab, set 'SENS_EN_MB12XX' to 1.

Once the sensor was connected to the Pixhawk and activated in QGroundControl, the bench tests could be performed. These tests included variations in material of the detected obstacle and angles. Figure 7.1a) demonstrates an experiment where the object to detect was in front of the sensor. In Fig. 7.1b), the idea was to determine the sonar capability of detecting an object which had an angular deflection (θ) in relation to the sensor.

(a) Straight on obstacle. (b) Off-set obstacle.

Figure 7.1: Ultrasonic sensor bench tests.

For these first experiments, the target object was a rectangular wooden board with size 30x25cm as seen in Fig. 7.2. The frontal test was also repeated with a rectangular XPS board (125.5x60cm) as the target object.

Figure 7.2: Experimental setup.

During these experiments, each board was positioned at different distances relative to the sonar (20cm to 760cm) and the sensor data was recorded for 30 seconds for each position. This method was aimed at determining, for each position, the fraction of time where the sensor was actually detecting its target and how much these measurements were deviated from the correct distance. All the sensor data collected during this period of time was stored in a ulog file format using the Pixhawk SD card. Then, these files were transferred to a computer, using the Log Download tool within the QGroundcontrol environment. Lastly, information from the original ulog file, including the detection rate and average absolute error, was extracted and analyzed using MATLAB.

Some aspects must be kept in consideration before displaying the empirical results. Given that these were acquired using a statistical approach, it is necessary to establish a minimal detection rate that is deemed acceptable for a given task. As a safety precaution, it was decided that, for this work, only a 100% detection rate would be sufficient to ensure that obstacles are in fact identified.

It is important to consider that the target material could affect the performance of this sensor. The MB1242 data sheet [61] mentions that this sensor's ideal surface to detect is hard, smooth and non-

porous. Although wood is not a perfect example of an ideal surface, its properties are not far from that category. To determine whether these conclusions are applicable to other materials, a target made of XPS was also tested, which is not as hard as wood, yet is more porous. Figure 7.3 represents the detection rate for both materials. When testing with a XPS board, the maximum range decreased and the sensor performed worse overall. This decrease in performance was foreseeable since XPS's properties do not match those of an ideal material (hard, smooth and non-porous).

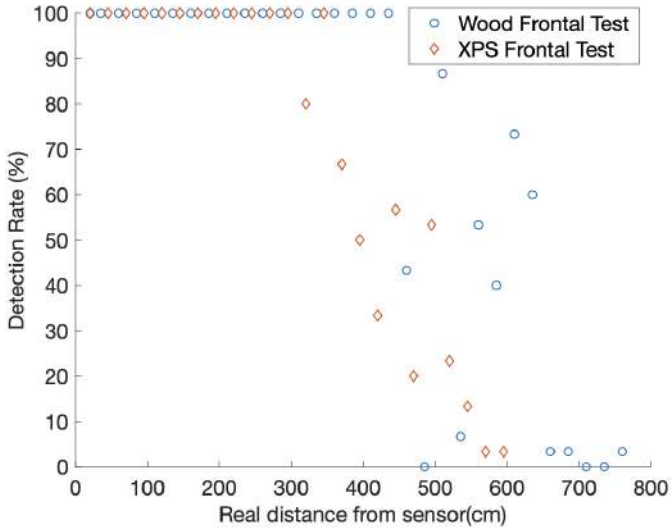


Figure 7.3: MB1242 detection rate for different materials.

The last significant point is that the performance of this sensor is impacted by the target's rotation within its inertial referential. Only when the sound is reflected back from the target can the sonar identify it. According to the principles of sound reflection, this is only conceivable if the normal vector of the surface in question is parallel to the trajectory of the sound being emitted until it reaches the desired target.

Empirically, this translates to the results that follow. Figure 7.4 shows the sensors detection rate for various distances and orientations. As expected, the sensor performed better when the obstacle was completely in front of it, achieving a maximum range of 435 cm with perfect a detection rate, although the datasheet states 640 cm. Additionally, the maximum range decreased when augmenting θ, which was also an expected behaviour. Moreover, this sensor proved to be very directional as it stopped detecting any targets for $\theta \geq 40°$.

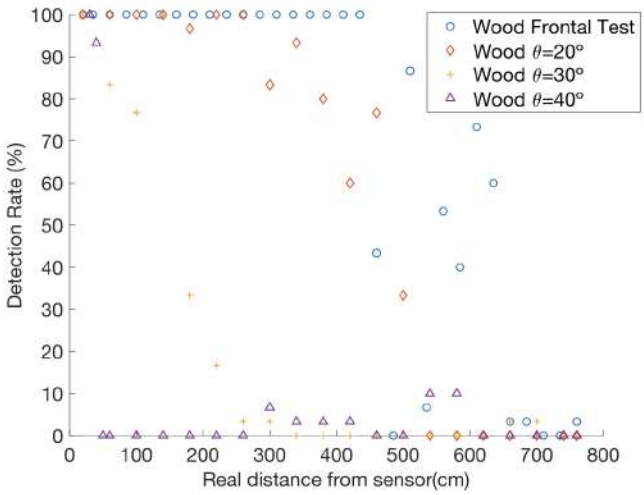


Figure 7.4: MB1242 detection rate for different angles of incidence.

Figure 7.5 shows the average absolute error at each distance from the sensor. No relation between the real distance from the sensor and its error was detected, given that all points seem to be almost randomly dispersed from 300 cm onwards. Moreover, the average absolute error never surpassed 30 cm. This error occurs in the wood frontal test, which leads to the conclusion that this board might not match the ideal surface for this type of sensor to detect. However, the collision avoidance system needs to be robust enough for most materials, *i.e.*, if the available sensor fails this bench test, it is not sufficiently accurate for the purpose of this work.

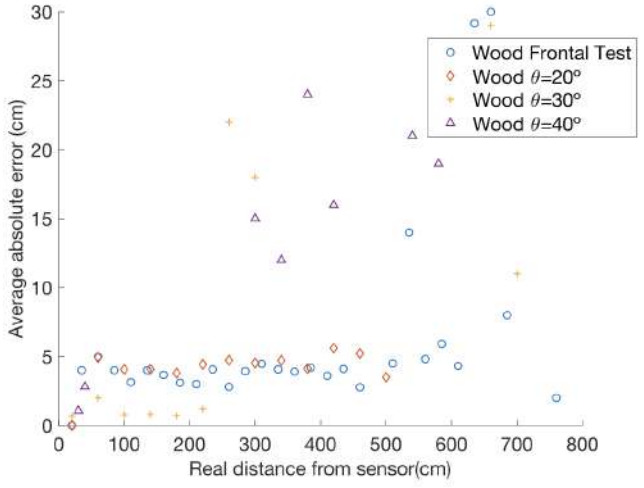
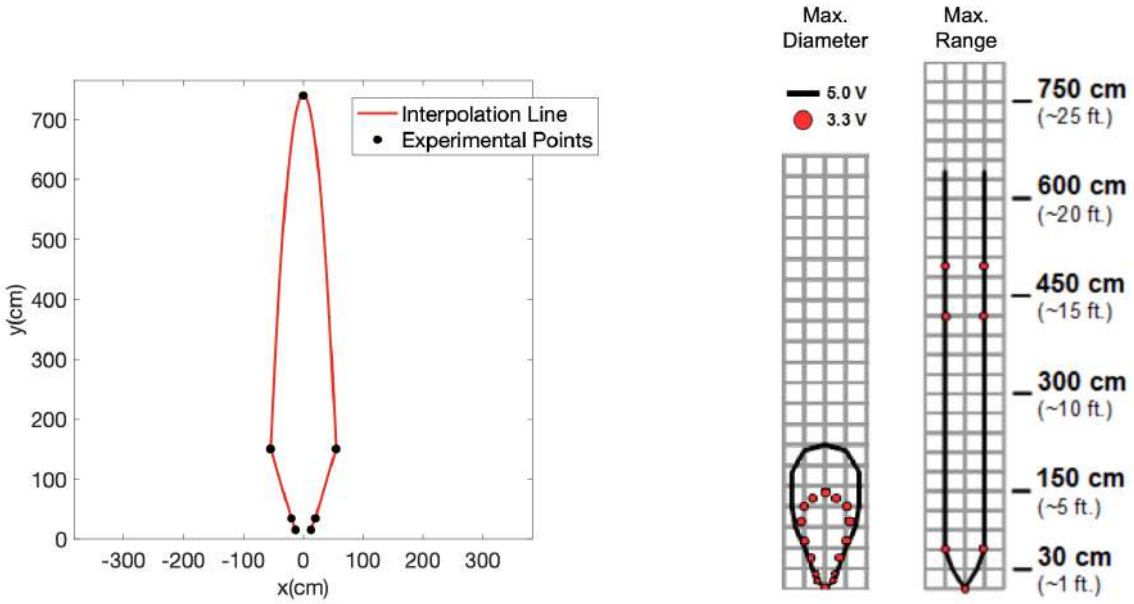


Figure 7.5: MB1242 average absolute error for different angles of incidence.

According to the datasheet [61], the sensor is calibrated and tested to provide stable range readings to large targets even in electrically and acoustically noisy environments. It also states that the sonar should ideally be used indoors. Nonetheless, it is important to note that these tests were done outdoors, due to the nature of this work. Since this sensor is intended to integrate a collision avoidance system for small UAVs, its applications require that it performs well outdoors. However, this also means that the results shown can and have been affected by exterior noise.

Additionally, an experimental beam pattern was generated using the maximum range recorded for each orientation, shown in Fig. 7.6. Such beam patterns tend to be particularly advantageous for S&A systems since they restrict their detecting volume, ultimately allowing the controlling device to pinpoint the target's location with a high degree of accuracy.



(a) Empirical. (b) Datasheet [57].

Figure 7.6: MB1242 beam pattern.

7.1.2 Laser Rangefinder

Prior to testing, it was necessary to configure the LW20/C laser rangefinder by LightWare [62] within the flight controller's environment. The LW20/C's I2C compatibility mode is not active by default. To activate it, is necessary to install and start Lightware Terminal. The sensor must then be connected to a computer using a serial to USB adapter (see Fig. 7.7). The LW20 will automatically be detected by the application, allowing settings to be altered and, in this case, 'Pixhawk I2C compatibility mode' to be activated.

Figure 7.7: LW20/C - USB adaptor connection. (source: Lightware [62].)

Once this task was executed, it was possible to activate this sensor within the QGroundControl environment. To do this, the parameter 'SENS_EN_SF1XX' was set to 'LW20/C'.

Finally, the perfomance of the sensor could be assessed through an identical experience to that of the sonar. However, since the laser rangefinder is completely directional, it was not necessary to experiment with off-set obstacles.

In frontal tests, the laser maintained a perfect detection rate before reaching 85 m, as seen in Fig. 7.8. From this distance onward, the detection rate decreased non-linearly until it reached 100 m (marked as a dashed red line in Fig. 7.8). Ultimately, the complete range promised in the datasheet was not attained with a perfect detection rate.

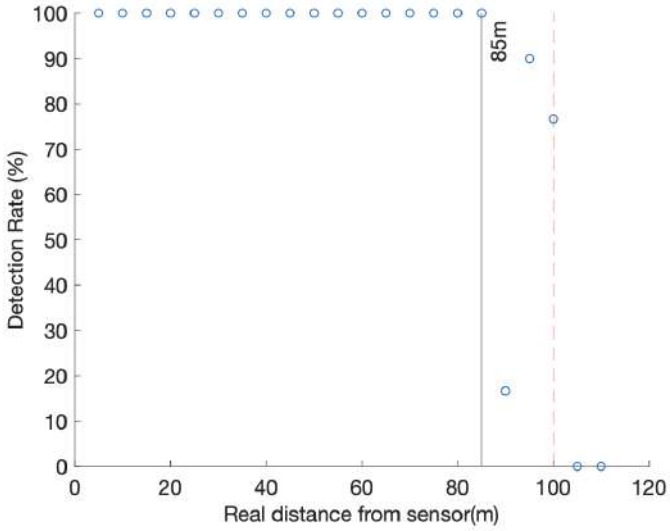


Figure 7.8: LW20/C detection rate.

These tests also served to prove how much of an impact directionality has on this type of sensor. The wooden board had to be perfectly aligned with the laser rangefinder in order for it to detect it correctly. When translating this to the optimal sensing system designed in Chap. 4, it means the sensor has to be flawlessly aligned with the UAV's longitudinal axis.

Lastly, the average absolute error, plotted in Fig. 7.9, was mostly between 0 and 50 cm, but increased with the distance from the sensor. The results were not as satisfactory for distances greater than 50 m.

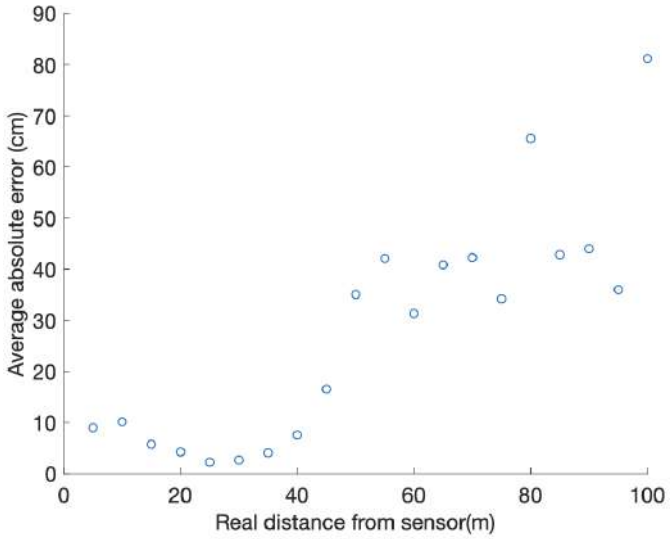


Figure 7.9: LW20/C average absolute error.

7.1.3 LIDAR

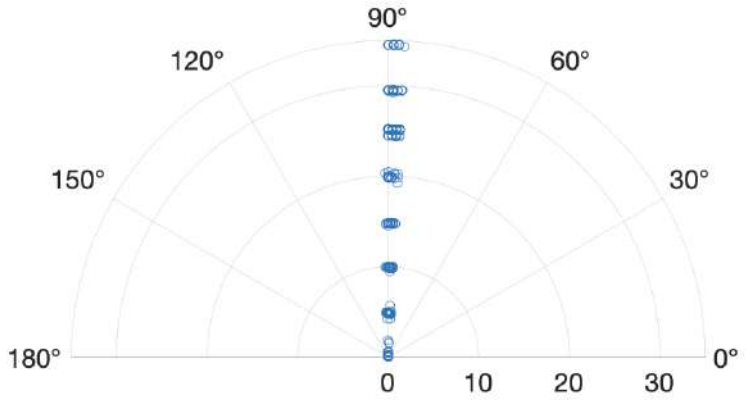
The chosen LIDAR was the SF45/B model by LightWare [63]. It comes with a micro USB port that connects to any PC running the LightWare Studio application for visualisation of results, making configuration changes and upgrading the firmware. The sensor was configured according to Tab. 7.1.

Table 7.1: LIDAR configuration parameters.

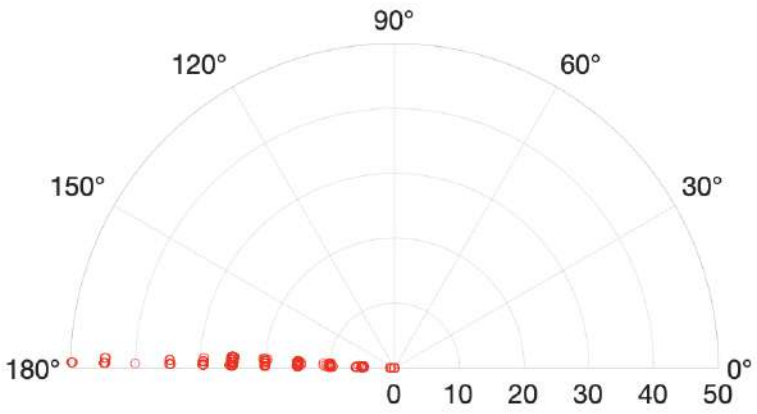
Low can angle limit	High scan angle limit	Baud rate	Update rate	Scan upon startup
-90°	+90°	921600	50 Hz	Enabled

To activate this sensor within the QGroundControl environment, it was necessary to follow the instructions in Sec. 6.1 to add the driver to firmware. It was then possible to access the vehicle setup section and, within the parameters tab, set SENS_EN_SF45_CFG to the desired serial port (TELEM2).

In the bench tests, the angles of the detected obstacle and scanning speed of the LIDAR were varied. Figure 7.10 shows the resulting scans from two different experiments: a) the object to detect is in front of the sensor; and b) the object to detect is at 90° in relation to the sensor. In these tests, the forementioned rectangular XPS board (125.5x60cm) was used as target.



(a) Straight on obstacle.



(b) 90° off-set obstacle.

Figure 7.10: LIDAR SF45/B bench tests.

On an unobstructed rugby field, the XPS board was positioned at different distances relative to the LIDAR (0 to 50 m) and the sensor data was recorded for 30 seconds for each position. As expected, this sensor performed better than the others, maintaining a perfect detection rate through all its range in both experiments, as seen in Fig. 7.11. However, Fig. 7.12 shows that the average absolute error was overall lower when the obstacle was aligned with the sensor. This is likely because the LIDAR scans back and forth from -90 to 90° , meaning that for each sweep, it passes twice through $\theta = 0^\circ$ and only once through each limit.

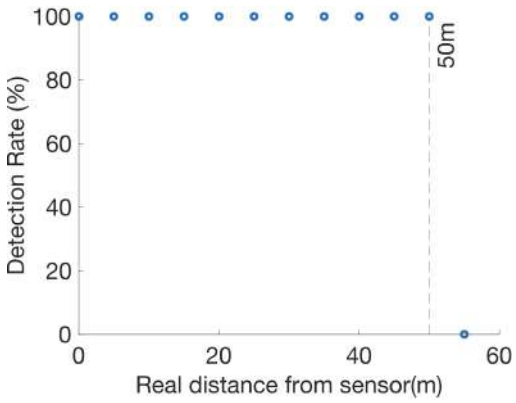


Figure 7.11: SF45/B detection rate.

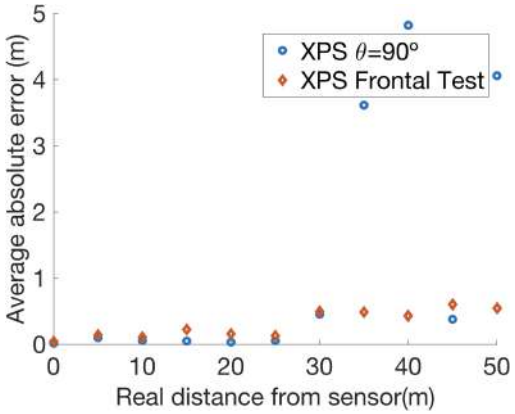


Figure 7.12: SF45/B average absolute error.

Although the SF45's update rate was set to 50Hz, empirically, it is, on average, 37.2Hz. In LightwareStudio, it is also possible to calibrate the sensor's cycle delay, which is inversely proportional to its scanning speed. The minimum cycle delay (5) corresponds to the maximum scanning speed (6.3 rad/s) and vice-versa. This implies that, by choosing a higher sweep speed and maintaining the angle limits, the arc of circle that is not being detected between each measurement increases. Figure 7.13 illustrates how the length of the arc traversed varies analytically with the distance to the sensor and the angular velocity. This graphic shows that, although the LIDAR has a 50 m range, at the maximum scanning speed, it might not be possible to detect an obstacle less than 8 m wide at this distance. When the scanning speed is reduced, the sensor is likely to detect a target of at least 2.2 m at maximum distance. At minimum speed, this stops being relevant within the 50 m range. However, if covering a larger area quickly is more important, sacrificing some visibility at the maximum range might be acceptable. Ultimately, the compromise should be based on the specific needs and constraints of the system.

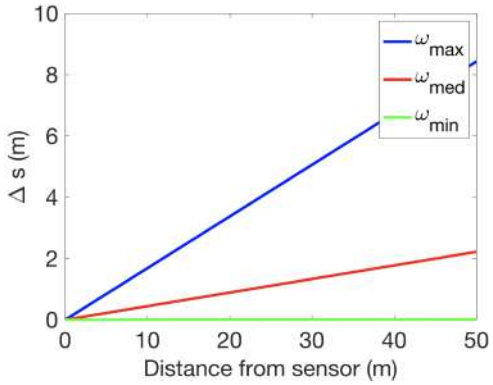


Figure 7.13: SF45/B undetectable arcs for different scanning speeds.

7.1.4 RADAR

The US-D1 RADAR by Ainstein [64] is not automatically included in most firmware, hence it cannot be used just by setting a parameter through QGroundControl (as was possible with the sonar and laser rangefinders). To use it, it is necessary to add the driver to firmware and update a configuration file to start the driver on boot, as mentioned in Sec. 6.1.

The sensor can be connected to any unused serial port (UART) by simply configuring the serial port on which the RADAR will run using 'SENS_ULAND_CFG' in the vehicle setup section. In this case, the parameter was set to TELEM2. There is no need to set the baud rate for the port, as this is configured by the driver.

After the initial calibration, the RADAR system first appeared to be functional; however, it subsequently encountered challenges in its interactions with QGroundControl. These issues were attributed to potential integration faults within PX4. Consequently, the planned bench tests for this particular sensor could not be conducted. In light of time constraints and the need to maintain focus on the integration of the remaining sensors within a rover system, it was determined that the research would no longer rely on the radar system's obstacle detection capabilities.

7.2 Rover Tests

To be able to implement the most promising detection solutions, the system needed to be tested on a small rover first. Although it was being idealized for a UAV, ground testing the current system was a convenient intermediate experiment, due to the risk associated with flight testing. The experiment to be performed consisted of directing the rover along a linear path, equipped with a forward-facing distance sensor, and strategically positioning an obstacle directly in its trajectory, as illustrated in Fig. 7.14.

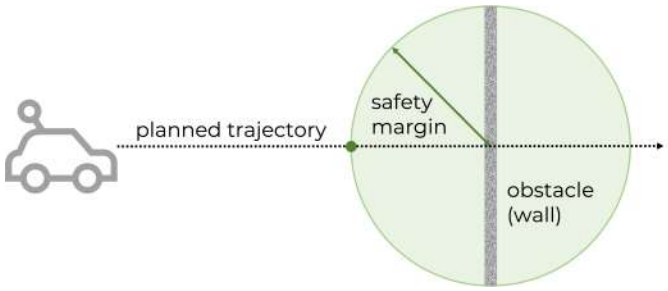


Figure 7.14: Rover proposed test.

7.2.1 Firmware Comparison and Rover Algorithms

As mentioned in Sec. 2.4.4, PX4 and Ardupilot are the leading autopilot systems for unmaned vehicles. Both UAVs and rovers fall into this category, however, Ardupilot is a more vehicle-specific firmware.

Although PX4 offers features that would suit this work, such as Obstacle Avoidance [55] and Collision Prevention [75], these are currently only developed for multicopter applications. Therefore, one possibility would be to further develop the PX4 code. Nonetheless, since the current system is being projected for UAVs, it is reasonable to simplify the rover experiments in particular by switching to Ardupilot.

Similar to the operational framework of PX4, the ArduRover code orchestrates a myriad of functions, each operating at predetermined frequencies, thereby enabling their concurrent execution while constantly uptading system data. These functions encompass a spectrum of tasks, such as sensor data acquisition, radio signal processing, telemetry signal exchange, and the real-time adaptation of servo outputs.

Among the realm of ArduRover functions, some go by the form 'Modexxx.update', where 'xxx' denotes the specific operational mode of the rover (e.g., Manual, Hold, Auto, Acro). These are invoked in the 'Rover.cpp' file with the primary objective of regulating the PWM signals that govern the steering servo and the motor ESC. It is noteworthy to highlight the distinguishing approaches adopted by these functions in pursuit of their shared objective.

For instance, the Modemanual.update() and Modehold.update() functions execute this task directly. The former interprets radio channel inputs from the Pixhawk, translating them into corresponding PWM signals for the steering servo and motor ESC. In contrast, the Hold mode strictly outputs PWM signals with a 0% duty cycle, effectively immobilizing the vehicle. On the other hand, all other modes call for an intermediary step to determine the desired speed and heading, subsequently calculating the throttle and steering values that align with the respective reference points. In Auto mode, this calculation considers the rover's current position, the next waypoint, and the desired cruising speed.

Once these drive modes ascertain the reference values for speed and heading, the corresponding steering and throttle is promptly calculated, unless the avoidance algorithm is activated. In the presence of obstacle avoidance, these modes evaluate the immediate surroundings, utilizing at least one distance sensor, and then make real-time adjustments to their speed and heading to ensure the rover's

safe traversal. For this particular study, the simplest obstacle avoidance algorithm was used, effecting a cautious interruption of motion when the system approaches a predefined safety threshold. Upon processing data from the distance sensor, the avoidance algorithm performs an assessment to determine whether there exists a potential risk of surpassing this safety margin. Subsequently, the algorithm proceeds to modify solely the target speed, instigating a gradual deceleration, as a preventive measure to avoid breaching the established safety margin.

7.2.2 Eletrical Layout

To test everything as a unique system, the first practical step is the hardware integration. A new eletrical diagram is proposed in Fig. 7.15 (as opposed to the one in Fig. 5.4).

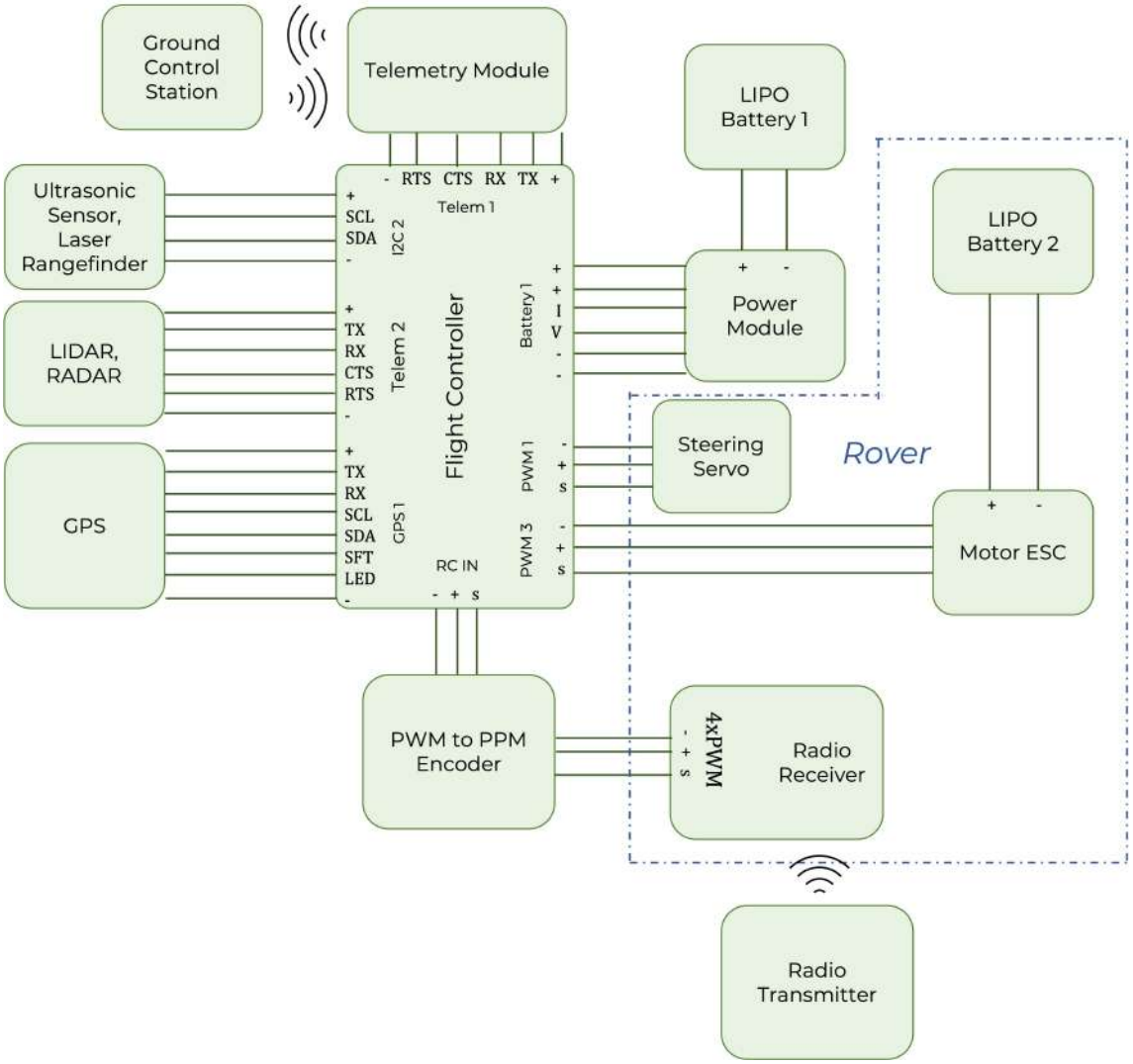


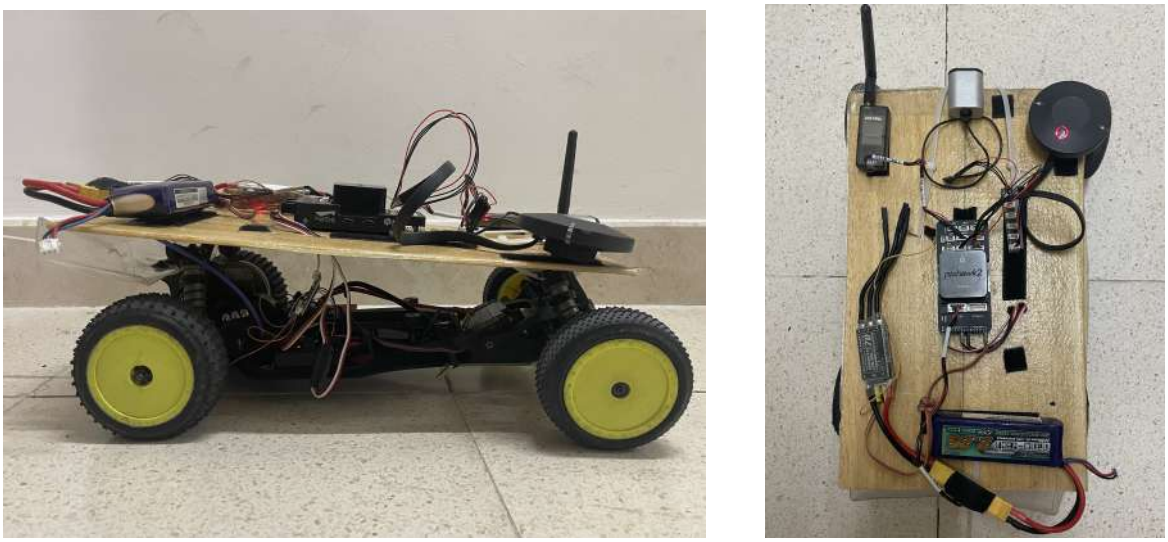
Figure 7.15: Electrical diagram for rover testing [20].

The components responsible for radio signal reception and PPM encoding have been restructured and a secondary battery has been introduced to cater solely to the rover's power requirements. This adjustment also yields an increase in the overall autonomy of the system.

The radio transmitter transmits all radio signals, which are subsequently intercepted by the radio receiver integrated into the rover. To facilitate the reception of these radio signals by the Pixhawk flight controller, a PPM encoder is employed. This encoder receives and consolidates the data from the four radio channels into a single PPM channel, which is then relayed to the Pixhawk through the RCIN port.

Two of these four channels are employed for manual rover control, with one channel governing steering and the other regulating throttle. One of the remaining channels was allocated for a supplementary function: switching from manual to auto mode. The Pixhawk controls the vehicle by transmitting PWM signals for throttle and steering to the motor ESC and the steering servo, respectively.

The implemented layout is depicted in Fig. 7.16. In this case, the rover was equipped with a laser.



(a) Side view. (b) Top view.

Figure 7.16: Assembly of the proposed layout (rover with laser-based collision avoidance system).

7.2.3 Software Configuration

Once the physical setup of the rover is configured, ArduPilot shall be installed onto the Pixhawk flight controller, as mentioned in Sec. 7.2.1. This steo is crucial for conducting subsequent tests to evaluate the rover's response to obstacles in its trajectory. The firmware setup process is carried out through QGroundControl, analogously to what was described in Sec. 6.2.

The radio calibration automatically changes the 'RCMAP_xxx' parameters to specify the desired channel number for each function (pitch, roll, throttle and yaw). In this case, the assignment of 'RCMAP_THROTTLE' is linked to channel 3, while 'RCMAP_ROLL' pertains to channel 4 and refers to the steering function. In the parameter editor, channel 2 can also be configured to alter the rover's drive mode by setting the parameter 'MODE_CH' to 2. Table 7.2 summarizes these values.

Table 7.2: Radio calibration parameters.

RCMAP_THROTTLE	RCMAP_ROLL	MODE_CH
3	4	2

Subsequently, several critical parameters related to the rover's performance shall be defined. These encompass the rover's cruise speed, cruise throttle, and maximum turn rate, allowing the firmware to effectively manage steering and throttle.

Regarding the simple avoidance algorithm, it must first be verified that each sensor is correctly set up by changing 'RNGFNDx_TYPE' to the respective rangefinder model (MaxbotixI2C or LightWareI2C). Additional parameters relative to the sensors range and scaling factor will also be required and can be found in each rangefinders' setup guide [76] [77]. If the sensor in use is the LIDAR, the parameters to adjust would be the ones in Tab. 7.3: 'SERIAL2_PROTOCOL' to 'Lidar360' and 'SERIAL2_BAUD' to '115' (when connecting the LIDAR to the Telem2 port) [78].

Table 7.3: LIDAR setup parameters in ArduRover.

SERIAL2_PROTOCOL	SERIAL2_BAUD
Lidar360	115

Once the sensors' setup is complete, the chosen rangefinder, its orientation, the desired safety margin, the activation status of the avoidance algorithm, and the preferred response of the rover (whether to initiate a simple stop or execute a more sophisticated avoidance maneuver) must also be specified. A summary of the most pivotal parameters, including their corresponding values, is presented in Tab. 7.4. This way, the rover attempts to stop the vehicle before it hits objects in all modes except manual.

Table 7.4: Firmware setup parameters.

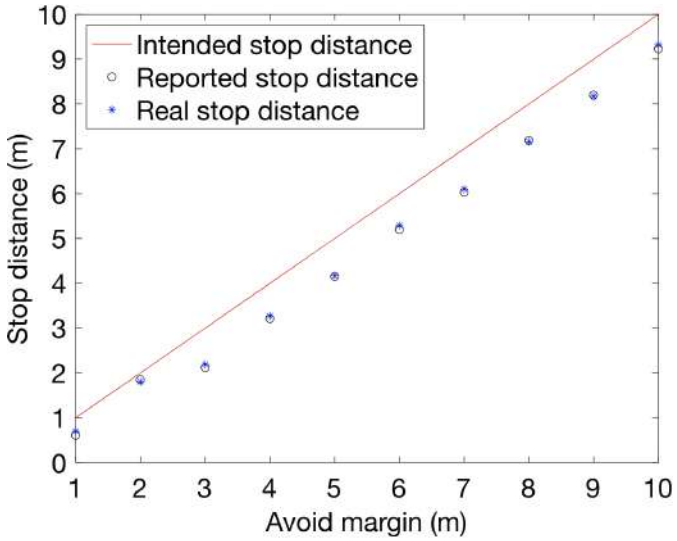
Rover motion		Collision avoidance				
CRUISE_THROTTLE	CRUISE_SPEED	RNGFND1_TYPE	PRX1_TYPE	OA_TYPE	AVOID_ENABLE	AVOID_ENABLE
10%	$1.00\ \mathrm{m/s}$	$7/2$	$6/9$	0	7	$[1\ \ 2\ \ 3\ \ \ldots\ \ 10]$

From left to right, these parameters dictate the throttle percentage required to attain the cruise speed, the actual cruise speed, the rangefinder type utilized (2 for MB1242 or 7 for LW20/C), the proximity sensor to use for obstacle avoidance (6 for RangeFinder and 9 SF45B) the specified object avoidance method (with 0 signifying a simple stop), and a switch that enables or disables the object avoidance functionality. In this case, 7 refers to 'All', meaning it uses all sources of barrier information, including proximity sensors. Finally, the last parameter is relative to the safety margin (in meters) that the rover must maintain from the obstacle.

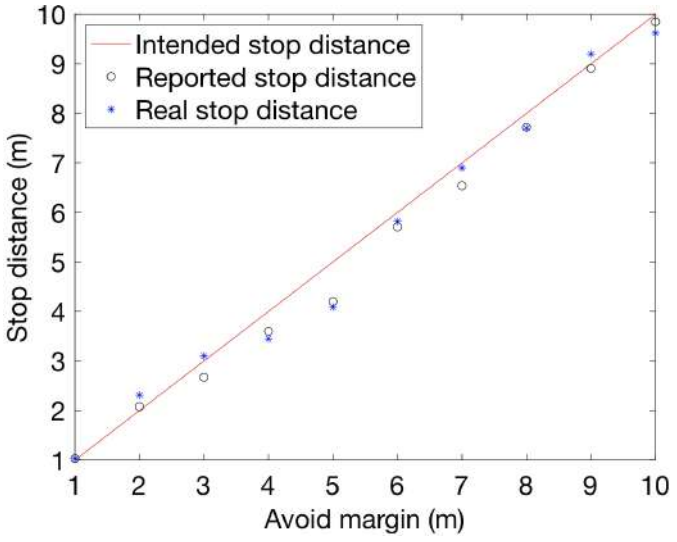
7.2.4 Results

After placing the rover on a collision path, the avoidance response was extracted and can be seen in Fig. 7.18. The tests were conducted by iterating on the 'AVOID_MARGIN' parameter (ranging from 1 to 10 meters in 1m increments), depicted in Fig. 7.17, and the reported and real distances at which the rover effectively stopped were registered.

Figure 7.17: Rover empirical test.



(a) Ultrasonic sensor MB1242.



(b) Laser rangefinder LW20/C.

Figure 7.18: Simple avoidance response with rangefinder mounted on rover.

This process was attempted using an ultrasonic sensor and laser rangefinder. The rover did not always succeed in stopping before the input safety margin, regardless of the sensor in use. However, it performed the best with the laser rangefinder, keeping a maximum margin of 80 cm from the intended stop distance. The sonar stop distances were more precise but less accurate, maintaining an average difference of 72 cm from the intended stop distance. This phenomenon, coupled with the fact that when equipped with the laser, the rover stopped before the intended stop distance, is likely due to the directionality of this type of rangefinder. In other words, as the laser is scanning the area, it might be detecting parts of the terrain that are irregular before it detects the wall (obstacle) and therefore commands to stop the rover are being sent earlier than necessary.

The system's imperfect performance can also derive from faulty calibration upon the definition of 'CRUISE_THROTTLE' and 'CRUISE_SPEED'. These are input values that can be changed directly by the user. Nonetheless, ArduRover also offers a funcionality that estimates the vehicle's cruise speed as the throttle command is manually being sent to the pixhawk and sets these parameters accordingly. Lacking a precise correlation between these two parameters, the performance of auto modes will be compromised since the controller will be unable to effectively maintain the rover's desired speed.

It is also worth mentioning that this type of test was attempted with a LIDAR too. However, the simple obstacle avoidance algorithm does not seem to respond with this type of sensors.

In summary, despite ArduRover's avoidance algorithm being functional with negligible error occurrence, the system has potential for more improvement and can be further tailored to suit this particular application.

Chapter 8

Conclusions

8.1 Achievements

This work presents a comprehensive solution for enhancing the safety of small fixed-wing UAVs by addressing the critical issue of obstacle detection during flight. A set of select sensors, namely the ultrasonic sensor, laser rangefinder, LIDAR, and RADAR, were identified and further employed in modeling collision detection and avoidance simulations using the potential fields method. Traditional Kalman filters were sufficient to provide proper tracking for laser rangefinders and LIDARs, but RADARs required a Converted Measurement Kalman Filter with unbiased conversion. Due to its simplicity and efficiency, the weighted filter technique was chosen at the decision level for the data fusion from many redundant sensors.

To determine the best combination of sensors and their orientations, simulations were used in an optimization study. The study revealed that relatively simple detection configurations can yield a high success rate in collision avoidance. While the ultrasonic sensor is found to be inadequate due to its limited range, the laser rangefinder benefits from a long range, but has a restricted field-of-view. On the other hand, both the LIDAR and RADAR prove to be the most promising options, offering not only a substantial range but also a wide field-of-view. Based on the optimization study, the recommended multi-sensor configurations consist of a front-facing LIDAR or RADAR, accompanied by a pair of laser rangefinders pointing sideways at either a 10 or 63 $^\circ$ angle.

To validate the proposed system, the necessary hardware and software were successfully implemented, which allowed for the individual testing of each sensor (except the RADAR). The bench tests confirmed the accuracy of the sensors specifications and previous simulations. In the case of the ultrasonic sensor, the importance of the material and the angular deflection of the obstacle to be detected was highlighted. As for the laser rangefinder, the key factor proved to be directionality. The LIDAR presented less shortcomings, as expected. However, the sensor's parameters (update rate, angular velocity and scan angle limits) directly affected its performance. More specifically, it is necessary to reach a compromise between the LIDAR scan speed and the effective range of visibility.

Lastly, another electrical layout was designed to facilitate the initial implementation of the system on

a small rover and to test its ability to avoid obstacles. Using an ultrasonic sensor and a laser rangefinder, the rover stopping distances were tested. The laser rangefinder consistently performed the best, maintaining at least an 80 cm margin from the intended stopping distance, while the ultrasonic sensor was more precise but stopped, on average, 72 cm away from the intended stopping distance. This behavior, possibly due to the laser's directionality detecting terrain irregularities before obstacles, highlights the importance of sensor choice. Additional inaccuracies can likely be linked to faulty calibration. Overall, while the avoidance algorithm exhibited satisfactory performance, it offers potential for further refinement to meet the specific needs of this particular application.

This work provides a comprehensive methodology for testing and validation of an optimized multi-sensor system, successfully attaining the primary goal of focusing on its detection capabilities. The proposed system shows great promise for enhancing the safety of small fixed-wing UAVs during flight.

8.2 Future Work

In the aftermath of this master's thesis, there are several promising avenues for future work and research. These endeavors aim to enhance the functionality of obstacle avoidance algorithms, especially within PX4 firmware.

Building upon the foundation established in this thesis, future work could delve into the development of PX4 sensor integration code. The primary focus would be on achieving a comprehensive connection with the LIDAR, ensuring that all the sensor's features can be effectively utilized and further employed in avoidance algorithms. Leveraging the sensor's scanning angle to detect the direction from which obstacles approach is a critical aspect to refine. Secondly, extending the work on sensor integration, the research must expand into multi-sensor integration and data fusion within PX4 firmware.

Once the code has been developed to seamlessly handle and combine data from various rangefinders, this feature can be employed in avoidance algorithms. Specifically, creating an avoidance algorithm tailored for rovers is a valuable pursuit, given that a transition to Ardupilot became imperative in the last stage of this work, driven by the inherent constraint of PX4's collision avoidance algorithms, which primarily cater to multicopters and lack robust support for rover applications.

Building upon the previous point, future work can involve the development and testing of advanced avoidance algorithms that encompass re-routing and path re-planning such as the potential fields method implemented in the simulation tool. While stopping a vehicle upon encountering an obstacle suffices for rovers, it is imperative to develop more sophisticated algorithms for UAVs that cannot be simply commanded to stop mid-flight. As the algorithms and codes mature, transitioning from rover implementation to UAVs is a logical progression. This entails adapting and optimizing the developed systems for UAV platforms, which have distinct operational requirements and challenges. The culmination of this work lies in the validation of the UAV system performance under realistic conditions.

Bibliography

[1] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif. Towards the Unmanned Aerial Vehicles (UAVs): A Comprehensive Review. *Drones*, 6(6):147, 2022.

[2] N. Muchiri and S. Kimathi. A Review of Applications and Potential Applications of UAV. In *Proceedings of the 2016 Annual Conference on Sustainable Research and Innovation*, Nairobi, Kenya, May 2016.

[3] H. A. Foudeh, P. C.-K. Luk, and J. F. Whidborne. An Advanced Unmanned Aerial Vehicle (UAV) Approach via Learning-Based Control for Overhead Power Line Monitoring: A Comprehensive Review. *IEEE Access*, 2021. doi:10.1109/ACCESS.2021.3110159.

[4] S. Berrahal, J.-H. Kim, S. Rekhis, N. Boudriga, D. Wilkins, and J. Acevedo. Border surveillance monitoring using Quadcopter UAV-Aided Wireless Sensor Networks. *Journal of Communications Software and Systems*, 12(1), March 2016. doi:10.24138/jcomss.v12i1.92.

[5] Commercial UAV Market Share, Size, Trends & Industry Analysis Report By Type; By End-Use; By Region; Segment Forecast, 2021 - 2028. *Polaris Market Research*, 2021. Acessed: 05/10/2022.

[6] K. Dalamagkidis. Classification of UAVs. pages 83–91, 2015. in Handbook of Unmanned Aerial Vehicles, doi:10.1007/978-90-481-9707-1_94.

[7] Tekever AR4. `https://www.tekever.com/models/ar4/`. Acessed: 05/10/2022.

[8] J. N. Yasin, M.-H. Haghbayan, M. M. Yasin, and J. Plosila. Swarm formation morphing for congestion-aware collision avoidance. *Helyion*, August 2021. doi:10.1016/j.heliyon.2021.e07840.

[9] MQ-4C Triton. `https://www.northropgrumman.com/what-we-do/air/triton/`. Acessed: 12/10/2022.

[10] DJI Guidance. `https://www.dji.com/pt/guidance`. Acessed: 05/10/2022.

[11] L. Tong, X. Gan, Y. Wu, N. Yang, and M. Lv. An ADS-B Information-Based Collision Avoidance Methodology to UAV. *Actuators*, 12(4), 2023. ISSN 2076-0825. doi:10.3390/act12040165.

[12] Next Generation Air Transportation System (NextGen). `https://www.faa.gov/nextgen`. Acessed: 10/10/2022.

[13] Title 14 Code of Federal Regulations (CFR) Part 91.113 and RTCA. https://www.govinfo.gov/content/pkg/CFR-2007-title14-vol1/html/CFR-2007-title14-vol1.htm, 2007. Acessed: 05/10/2022.

[14] Unmanned Aircraft Systems Beyond Visual Line of Sight Aviation Rulemaking Committee. Final Report. Technical report, Federal Aviation Administration, USA, March 2022.

[15] M. Grote, A. Pilko, J. Scanlan, T. Cherrett, J. Dickinson, A. Smith, A. Oakey, and G. Marsden. Sharing airspace with Uncrewed Aerial Vehicles (UAVs): Views of the General Aviation (GA) community. *Journal of Air Transport Management*, 102:102218, April 2022. doi:10.1016/j.jairtraman.2022.102218.

[16] F. Škultéty, K. Šajbanová, M. Janovec, and J. Rostáš. Unmanned Aircraft Systems on the Up: The Comparison between UK and US Drone Safety Issues. In *11th International Conference on Air Transport – INAIR 2022, Returning to the Skies*, volume 65, pages 361–367, Zilina, Slovakia, 2022. doi:10.1016/j.trpro.2022.11.040.

[17] E. Fakhraian, I. Semanjski, S. Semanjski, and E.-H. Aghezzaf. Towards Safe and Efficient Unmanned Aircraft System Operations: Literature Review of Digital Twins Applications and European Union Regulatory Compliance. *Drones*, 7(7), 2023. doi:10.3390/drones7070478.

[18] European Commission. New EU rules on dedicated airspace for drones enter into force. https://transport.ec.europa.eu/news-events/news/new-eu-rules-dedicated-airspace-drones-enter-force-2023-01-26_en, January 2023. Acessed: 24/10/2023.

[19] N. Alturas. Modeling and Optimization of an Obstacle Detection System for Small UAV's. Master's thesis, Instituto Superior Técnico, Lisboa, Portugal, January 2021.

[20] P. Serrano. Optimization of Obstacle Detection for Small UAVs. Master's thesis, Instituto Superior Técnico, Lisboa, Portugal, June 2022.

[21] S. Lin, X. Kong, and L. Liu. Development of an intelligent UAV path planning approach to minimize the costs in flight distance, time, altitude, and obstacle collision . In *2019 19th International Symposium on Communications and Information Technologies (ISCIT)*, Ho Chi Minh City, Vietnam, 2019. doi:10.1109/ISCIT.2019.8905119.

[22] P. Krishnan and K. Manimala. Implementation of optimized dynamic trajectory modification algorithm to avoid obstacles for secure navigation of UAV. *Applied Soft Computing Journal*, 90, January 2019. doi:10.1016/j.asoc.2020.106168.

[23] J.-y. Zhuang, L. Zhang, S.-q. Zhao, J. Cao, B. Wang, and H.-b. Sun. Radar-based collision avoidance for unmanned surface vehicles. *China Ocean Engineering*, 30(6), 2016. doi:10.1007/s13344-016-0056-0.

[24] C. Almeida, T. Franco, H. Ferreira, A. Martins, R. Santos, J. Almeida, J. Carvalho, and E. Silva. Radar based collision detection developments on USV ROAZ II. In *OCEANS 2009-EUROPE*, pages 1 – 6, Bremen, Germany, May 2009. doi:10.1109/OCEANSE.2009.5278238.

[25] J. Han, Y. Cho, J. Kim, J. Kim, N. Son, and S. Y. Kim. Autonomous collision detection and avoidance for aragon usv: Development and field tests. *Journal of Field Robotics*, 37(6):987–1002, 2020. doi:110.1002/rob.21935.

[26] A. Sorbara, E. Zereik, M. Bibuli, G. Bruzzone, and M. Caccia. Low cost optronic obstacle detection sensor for unmanned surface vehicles. In *2015 IEEE Sensors Applications Symposium (SAS)*, volume 10, pages 1–6, Zadar, Croatia, 2015. doi:10.1109/SAS.2015.7133652.

[27] P. Wang, Y. Wang, X. Wang, Y. Liu, and J. Zhang. An Intelligent Actuator of an Indoor Logistics System Based on Multi-Sensor Fusion. *Actuators*, 10(6), June 2021. doi:10.3390/act10060120.

[28] L. Nobile, M. Randazzo, M. Colledanchise, L. Monorchio, W. Villa, F. Puja, and L. Natale. Active Exploration for Obstacle Detection on a Mobile Humanoid Robot. *Actuators*, 10(9), June 2021. doi:10.3390/act10090205.

[29] L. Wang, R. Li, Z. Huangfu, Y. Feng, and Y. Chen. A Soft Actor-Critic Approach for a Blind Walking Hexapod Robot with Obstacle Avoidance. *Actuators*, 12(10), October 2023. doi:10.3390/act12100393.

[30] S. Karam, F. Nex, B. T. Chidura, and N. Kerle. Microdrone-Based Indoor Mapping with Graph SLAM. *Drones*, 6(11), November 2022. doi:10.3390/drones6110352.

[31] E. Aldao, L. M. Gonzalez-de Santos, and H. Gonzalez-Jorge. LiDAR Based Detect and Avoid System for UAV Navigation in UAM Corridors. *Drones*, 6(8), July 2022. doi:10.3390/drones6080185.

[32] H.-A. Langaker, H. Kjerkreit, C. L. Syversen, R. J. Moore, O. H. Holhjem, I. Jensen, A. Morrison, A. A. Transeth, O. Kvien, G. Berg, T. A. Olsen, A. Hatlestad, T. Negard, R. Broch, and J. E. Johnsen. An autonomous drone-based system for inspection of electrical substations. *International Journal of Advanced Robotic Systems*, 18(2), April 2021. doi:10.1177/17298814211002973.

[33] M. Mugnai, M. Teppati Lose, E. P. Herrera-Alarcon, G. Baris, M. Satler, and C. A. Avizzano. An Efficient Framework for Autonomous UAV Missions in Partially-Unknown GNSS-Denied Environments. *Drones*, 7(7), July 2023. doi:10.3390/drones7070471.

[34] B. Harvey and S. OYoung. Acoustic Detection of a Fixed-Wing UAV. *Drones*, 2(1), January 2018. doi:10.3390/drones2010004.

[35] M. Skowron, W. Chmielowiec, K. Glowacka, M. Krupa, and A. Srebro. Sense and avoid for small unmanned aircraft systems: Research on methods and best practices. *Journal of Aerospace Engineering*, 233(16), June 2019. doi:10.1177/0954410019867802.

[36] L. Yang, X. Feng, J. Zhang, and X. Shu. Multi-ray modeling of ultrasonic sensors and application for micro-UAV localization in indoor environments. *Sensors*, 19(8):1770, 2019. doi:10.3390/s19081770.

[37] M. Schirrmann, A. Hamdorf, A. Giebel, F. Gleiniger, M. Pflanz, and K.-H. Dammer. Regression kriging for improving crop height models fusing ultra-sonic sensing with UAV imagery. *Remote Sensing*, 9(7):665, 2017. doi:10.3390/rs9070665.

[38] D. G. Davies, R. C. Bolam, Y. Vagapov, and P. Excell. Ultrasonic sensor for UAV flight navigation. In *2018 25th International Workshop on Electric Drives: Optimization in Control of Electric Drives (IWED)*, Moscow, Russia, 2018. IEEE. doi:10.1109/IWED.2018.8321389.

[39] J. Saunders, B. Call, A. Curtis, R. Beard, and T. McLain. Static and dynamic obstacle avoidance in miniature air vehicles. In *Infotech@ Aerospace*, page 6950. Provo, UT, USA, September 2005.

[40] F. Wang, J. Cui, S. K. Phang, B. M. Chen, and T. H. Lee. A mono-camera and scanning laser range finder based UAV indoor navigation system. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 694–701, Atlanta, GA, USA, 2013. doi:10.1109/ICUAS.2013.6564750.

[41] D. Yin and L. Wang. Individual mangrove tree measurement using UAV-based LiDAR data: Possibilities and challenges. *Remote Sensing of Environment*, 223:34–49, 2019. doi:10.1016/j.rse.2018.12.034.

[42] A. F. Scannapieco, A. Renga, G. Fasano, and A. Moccia. Ultralight radar sensor for autonomous operations by micro-UAS. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, USA, 2016. p. 727-735. doi:10.1109/ICUAS.2016.7502664.

[43] M. Schartel, R. Burr, W. Mayer, N. Docci, and C. Waldschmidt. UAV-based Ground Penetrating Synthetic Aperture Radar. In *2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, Munich, Germany, 2018. IEEE. doi:10.1109/ICMIM.2018.8443503.

[44] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005. doi:10.1109/TRO.2004.838026.

[45] B. Frouzandeh, S. E. Mahmoudi, A. A. Bitaghsir, and A. Marandi. Mobile Robot Navigation Control in Moving Obstacle Environment Using Genetic Algorithm, Artificial Neural Networks and A* Algorithm. In *2009 WRI World Congress on Computer Science and Information Engineering*, volume 4, pages 705–713, Los Angeles, CA, USA, 2009. doi:10.1109/CSIE.2009.854.

[46] Extended Kalman Filter Navigation Overview and Tuning. `https://ardupilot.org/dev/docs/extended-kalman-filter.html`. Acessed: 27/10/2022.

[47] I. Ulrich and J. Borenstein. VFH*: Local Obstacle Avoidance with Look-Ahead Verification. In *2000 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2505–2511, San Francisco, CA, USA, 2000. IEEE. doi: 10.1109/ROBOT.2000.846405.

[48] I. Ulrich and J. Borenstein. VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots. In *1998 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1572–1577 vol.2, Leuven, Belgium, 1998. IEEE. doi:10.1109/ROBOT.1998.677362.

[49] S. Vanneste, B. Bellekens, and M. Weyn. 3DVFH+: Real-time three-dimensional obstacle avoidance using an Octomap. In *MORSE 2014 Model-Driven Robot Software Engineering: proceedings of the 1st International Workshop on Model-Driven Robot Software Engineering co-located with International Conference on Software Technologies: Applications and Foundations (STAF 2014)*, number 1319, pages 91–102, York, UK, 2014.

[50] N. Alturas and A. Marta. Modeling and Optimization of an Obstacle Detection System for Small Fixed-wing UAV. In *Aerobest 2021 - ECCOMAS Thematic Conference on Multidisciplinary Design Optimization of Aerospace Systems*, Lisboa, Portugal, 2021. ISBN:978-989-99424-8-6.

[51] H. Safadi. Local Path Planning Using Virtual Potential Field. `https://www.cs.mcgill.ca/~hsafad/robotics/`, 2007. Acessed: 04/11/2022.

[52] L. R. Ribeiro and N. M. F. Oliveira. UAV autopilot controllers test platform using Matlab/Simulink and X-Plane. In *2010 IEEE Frontiers in Education Conference (FIE)*, pages S2H–1–S2H–6, Arlington, VA, USA, 2010. doi:10.1109/FIE.2010.5673378.

[53] J. Leško, M. Schreiner, D. Megyesi, and L. Kovács. Pixhawk PX-4 Autopilot in Control of a Small Unmanned Airplane. In *2019 Modern Safety Technologies in Transportation (MOSATT)*, pages 90–93, Kosice, Slovakia, 2019. doi:10.1109/MOSATT48908.2019.8944101.

[54] A. Allouch, O. Cheikhrouhou, A. Koubâa, M. Khalgui, and T. Abbes. MAVSec: Securing the MAVLink Protocol for Ardupilot/PX4 Unmanned Aerial Systems. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 621–628, Tangier, Morocco, 2019. doi:10.1109/IWCMC.2019.8766667.

[55] PX4 Obstacle Avoidance. `https://docs.px4.io/v1.9.0/en/computer_vision/obstacle_avoidance.html`. Acessed: 27/10/2022.

[56] Mission Planner Home. `https://ardupilot.org/planner/`. Acessed: 24/11/2022.

[57] MaxBotix. `https://www.maxbotix.com/`. Acessed: 05/01/2023.

[58] M. Longbin, S. Xiaoquan, Z. Yiyu, S. Z. Kang, and Y. Bar-Shalom. Unbiased converted measurements for tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):1023–1027, 1998. doi:10.1109/7.705921.

[59] N. Gageik, P. Benz, and S. Montenegro. Obstacle Detection and Collision Avoidance for a UAV With Complementary Low-Cost Sensors. *IEEE Access*, 3:599–609, 2015. doi:10.1109/ACCESS.2015.2432455.

[60] International Civil Aviation Organization. *Rules of the Air, Annex 2 to the Convention on International Civil Aviation*. July 2005.

[61] MaxBotix I2CXL-MaxSonar-EZ Datasheet. https://maxbotix.com/pages/i2cxl-maxsonar-ez-datasheet. Acessed: 13/04/2023.

[62] LightWare LW20 LiDAR sensor Datasheet. https://www.documents.lightware.co.za/LW20%20-%20LiDAR%20Manual%20-%20Rev%2012.pdf. Acessed: 13/04/2023.

[63] LightWare SF45/B product guide. https://support.lightware.co.za/sf45b/#/specs. Acessed: 13/04/2023.

[64] Ainstein US-D1 Data Sheet. https://ainstein.ai/wp-content/uploads/US-D1-Data-Sheet.pdf. Acessed: 13/04/2023.

[65] J. Mankar, C. Darode, K. Trivedi, M. Kanoje, and P. Shahare. Review of I2C protocol. *International Journal of Research in Advent Technology*, 2(1), 2014.

[66] E. Peña and M. G. Legaspi. UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter. *Visit Analog*, 54(4), 2020.

[67] PixHawk Cube Black Flight Controller. https://ardupilot.org/copter/docs/common-thecube-overview.html, . Acessed: 17/04/2023.

[68] The Cube Module Overview. https://docs.cubepilot.org/user-guides/autopilot/the-cube-module-overview, . Acessed: 29/10/2023.

[69] Visual Studio Code. https://code.visualstudio.com/. Acessed: 20/05/2023.

[70] Building PX4 Software. https://docs.px4.io/main/en/dev_setup/building_px4.html, . Acessed: 20/05/2023.

[71] PX4 Board Configuration (Kconfig). https://docs.px4.io/main/en/hardware/porting_guide_config.html, . Acessed: 20/05/2023.

[72] F. Malacarne. PX4 Autopilot Customization for Non-standard Gimbal and UWB Peripherals. Master's thesis, Politecnico di Torino, Torino, Italia, 2020.

[73] PX4 UORB Explained : Part 4, ULog Flight Logging System. https://px4.io/px4-uorb-explained-part-4-ulog-flight-logging-system/. Acessed: 02/10/2023.

[74] QGroundControl User Guide. https://docs.qgroundcontrol.com/master/en/index.html. Acessed: 25/01/2023.

[75] PX4 Collision Prevention. https://docs.px4.io/main/en/computer_vision/collision_prevention.html. Acessed: 27/10/2022.

[76] Maxbotix I2C Sonar Rangefinder. https://ardupilot.org/copter/docs/common-rangefinder-maxbotixi2c.html, . Acessed: 12/10/2023.

[77] LightWare SF20 / LW20. https://ardupilot.org/copter/docs/common-lightware-lw20-lidar.html, . Acessed: 12/10/2023.

[78] LightWare SF45/B 350 Lidar. https://ardupilot.org/copter/docs/common-lightware-sf45b.html, . Acessed: 12/10/2023.

Appendix A

PX4 code for LIDAR SF45/B driver

```
PX4-Autopilot/src/drivers/distance_sensor/lightware_sf45_serial/lightware_sf45_serial.cpp
(...)
int SF45LaserSerial::collect()
{
(...)
  } else {
    ret = ::read(_fd, &readbuf[0], 10);
    uint8_t flags_payload = (readbuf[1] >> 6) | (readbuf[2] << 2);

    if (readbuf[3] == SF_DISTANCE_DATA_CM && flags_payload == 5) {
      for (uint8_t i = 0; i < ret; ++i) {
        sf45_request_handle(ret, readbuf);
      }
      if (_init_complete) {
        sf45_process_replies(&distance_m, &yaw_deg);
      } // end if
    } else {
      ret = ::read(_fd, &readbuf[0], 10);
    }
  }
  (...)
  PX4_DEBUG("val (float): %8.4f, raw: %s, valid: %s", (double)distance_m, _linebuf, ((
    _crc_valid) ? "OK" : "NO"));
  PX4_INFO("\n MARTA %llu DIST: %8.4f, YAW: %8.4f, valid: %s \n", timestamp_sample, (
    double)distance_m, (double)yaw_deg,((_crc_valid) ? "OK" : "NO"));
  _px4_rangefinder.update(timestamp_sample, yaw_deg);
  // Functional but wrong
  /*_px4_rangefinder.s_update(timestamp_sample, distance_m, yaw_deg);*/

  perf_end(_sample_perf);

  return PX4_OK;
}
(...)
```

```
void SF45LaserSerial::sf45_process_replies(float *distance_m, float *yaw_deg)
{
  switch (rx_field.msg_id) {
  case SF_DISTANCE_DATA_CM: {
      uint16_t obstacle_dist_cm = 0;
      const float raw_distance = (rx_field.data[0] << 0) | (rx_field.data[1] << 8);
      int16_t raw_yaw = ((rx_field.data[2] << 0) | (rx_field.data[3] << 8));
      int16_t scaled_yaw = 0;
      // The sensor scans from 0 to -160, so extract negative angle from int16 and
    represent as if a float
      if (raw_yaw > 32000) {
        raw_yaw = raw_yaw - 65535;
      }
      // The sensor is facing downward, so the sensor is flipped about it's x-axis -
    inverse of each yaw angle
      if (_orient_cfg == 1) {
        raw_yaw = raw_yaw * -1;
      }
      switch (_yaw_cfg) {
      case 0:
        break;
      case 1:
        if (raw_yaw > 180) {
          raw_yaw = raw_yaw - 180;
        } else {
          raw_yaw = raw_yaw + 180; // rotation facing aft
        }
        break;
      case 2:
        raw_yaw = raw_yaw + 90; // rotation facing right
        break;
      case 3:
        raw_yaw = raw_yaw - 90; // rotation facing left
        break;
      default:
        break;
      }
      scaled_yaw = raw_yaw * SF45_SCALE_FACTOR;
      *yaw_deg = scaled_yaw;
      // Convert to meters for rangefinder update
      *distance_m = raw_distance * SF45_SCALE_FACTOR;
      obstacle_dist_cm = (uint16_t)raw_distance;
      (...)
      _obstacle_distance_pub.publish(_obstacle_map_msg);
      break;
    }
  default:
    // add case for future use
    break;} } (...)
```

```
/Users/martaportugal/PX4-Autopilot/src/lib/drivers/rangefinder/PX4Rangefinder.cpp
(...)
void PX4Rangefinder::s_update(const hrt_abstime &timestamp_sample, const float distance,
    const float yaw, const int8_t quality)
{
  distance_sensor_s &report = _distance_sensor_pub.get();

  report.timestamp = timestamp_sample;
  report.current_distance = distance;
  report.current_yaw = yaw;
  report.signal_quality = quality;

  // if quality is unavailable (-1) set to 0 if distance is outside bounds
  if (quality < 0) {
    if ((distance < report.min_distance) || (distance > report.max_distance)) {
      report.signal_quality = 0;
    }
  }

  _distance_sensor_pub.update();
}
```