TÉCNICO LISBOA

LiDAR and Camera Sensor Fusion for Onboard sUAS Detection and Tracking

Daniel Alexandre da Silva Justino

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Afzal Suleman
Prof. Rodrigo Martins de Matos Ventura

Examination Committee

Chairperson: Prof. Paulo Jorge Coelho Ramalho Oliveira
Supervisor: Prof. Afzal Suleman
Member of the Committee: Prof. Alexandra Bento Moutinho

January 2021

To my family, and to Rubinho, for everything

Acknowledgments

I want to thank Professor Afzal Suleman for the opportunity of doing my Master's thesis in such a wonderful place as Canada. It was a life-changing experience, and one I am truly grateful for.

I would like to also thank Professor Rodrigo Ventura for the advice and critique of my work that helped me grow and improve.

Additionally, an enormous thank you to António Ramalho for the countless hours and the eagerness to always help me. My most profound gratitude for his advice from the beginning to the end.

I want to thank my friends from Técnico: Tiago, Diogo, Sara, Gonçalo, Inês and Hélder, who came with me on this journey to Canada, for all the fantastic moments and fond memories we have created and which I will always remember.

I want to thank everybody at CfAR for the excellent guidance and support provided through all my time there.

I want to thank my best friends: Afonso, André and Miguel, for being like my brothers and bringing out the best in me.

I want to thank my parents and brother, for giving me everything and making me who I am. To my aunt Rosária, for always being there for me. To all my family for the unconditional support.

Finally, I want to thank my girlfriend, Filipa, for her love, support, affection, and for being my north star.

Resumo

O aumento do número de pequenos sistemas aéreos não tripulados (sUAS) tem criado preocupações perante organizações governamentais e militares, dado que estas podem comprometer infraestruturas e ameaçar aeronaves tripuladas. Detetar e seguir aeronaves não cooperativas é essencial para construir uma solução anti-sUAS eficaz. Sensores de bordo tipicamente envolvem câmaras eletro-óticas (EO), podendo ser leves e capturar informação de alta resolução. No entanto, a procura de alvos neste contexto é computacionalmente dispendiosa e suscetível à captura de falsos positivos. Além disso, estes sensores não conseguem medir distâncias diretamente, algo que um sensor LiDAR é capaz, produzindo nuvens de pontos com uma frequência até $20Hz$ e um alcance superior a $100m$. Contudo, dada a sua dispersão, estas nuvens não conseguem reconhecer pequenos alvos. Esta tese desenvolve um *YOLO-based tracker* para detetar e seguir visualmente sUAS e estuda a capacidade de um LiDAR para detetar sUAS a bordo de um multi-rotor. Adicionalmente, demonstra um procedimento de calibração extrínseco capaz de projetar pontos 3D do LiDAR para uma imagem de forma exata. A fusão sensorial proposta tem como objetivo criar regiões de interesse (ROI), obtidas através de deteções projetadas pelo LiDAR, para limitar a janela de procura do *YOLO-based tracker*. Finalmente, foram realizadas experiências com múltiplos alvos a bordo de uma aeronave, demonstrando que a fusão sensorial melhora a precisão do *YOLO-based tracker* de 17.0% para 91.2%, assim como a velocidade de processamento de $24Hz$ para $57Hz$, mantendo valores de sensibilidade semelhantes, 41.9% comparado a 48.4%.

Palavras-chave: YOLO-based tracker, regiões de interesse, calibração extrínseca, sistemas aéreos, sUAS

Abstract

The recent growth in numbers of small unmanned aerial systems (sUAS) has raised concerns among civilian and military organizations, as they can jeopardize critical infrastructure and threaten manned aircraft. Detecting and tracking intrusive drones is essential to construct reliable counter sUAS (C-sUAS) solutions. Onboard payload sensors typically include electro-optical (EO) cameras, which can be lightweight and provide high-resolution information of the surrounding scene. However, continually searching for targets across high-resolution images is computationally expensive and susceptible to an increase in false positives. Furthermore, EO cameras cannot measure distances directly, which light detection and ranging (LiDAR) sensors can, generating point clouds at a frequency up to $20Hz$ with ranges over $100m$. However, these are usually sparse and cannot recognize small targets. The present thesis studies each sensor's capabilities and develops a sensor fusion solution. It develops a *YOLO-based tracker* for visual detection and tracking and studies the ability of a LiDAR to detect sUAS onboard an aircraft. Additionally, it demonstrates an extrinsic calibration procedure to project 3D LiDAR points into the camera frame accurately. The proposed sensor fusion solution aims to create regions of interest (ROI) from these LiDAR projections to narrow the YOLO-based tracker's search window. Finally, experiments with multiple flying targets were performed onboard an aircraft, demonstrating that the sensor fusion solution improves the *YOLO-based tracker* baseline results, increasing precision from 17.0% to 91.2%, and framerate, from $24Hz$ to $57Hz$, keeping a similar recall of 41.9%, compared to 48.4%.

Keywords: YOLO-based tracker, regions of interest, extrinsic calibration, aerial systems, sUAS

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α	Elevation angle.
γ	Aspect ratio.
ω	Azimuth angle.
θ	Pitch angle of sensor system.

Roman symbols

d	Distance.
h	Altitude.
u	Component of the bounding box centre in the X-axis.
v	Component of the bounding box centre in the Y-axis.
L	Bounding box height.

Subscripts

i	Index of LiDAR horizontal scan.
j	Object class in dataset for object detection.
l	Index of target inside dataset.
max	Maximum value.
n	Point inside precision-recall curve.
t	Index of frame inside dataset.
s	Variable corresponding to the sensor system.
x, y, z	Cartesian components.

Superscripts

T	Transpose.

$\rightarrow$ Vector.

$\bullet$ First time derivative.

Glossary

CAD	Computer-aided design.
CAGR	Compound annual growth Rate.
CfAR	Centre for Aerospace Research.
CG	Centre of gravity.
CNN	Convolutional neural network.
CPU	Central processing unit.
C-sUAS	Counter small unmanned aerial systems.
FAA	Federal Aviation Administration.
FOV	Field of view.
GNSS	Global Navigation Satellite System.
GPU	Graphics processing unit.
ICAO	International Civil Aviation Organization.
IMU	Inertial measurement unit.
LiDAR	Light detection and ranging.
LSS	Low, slow and small.
NATO	North Atlantic Treaty Organization.
PnP	Perspective-n-point.
RCS	Radar cross-section.
ROI	Region of interest.
ROS	Robotic operating system.
RPA	Remotely piloted aircraft.
SNR	Signal-to-noise ratio.
SSH	Secure Socket Shell.
sUAS	Small unmanned aerial system.
TOF	Time of flight.
USB	Universal serial bus.
USD	United States Dollar.
VTOL	Vertical take-off and landing.

Chapter 1

Introduction

It has become easier than ever to acquire and operate Unmanned Aerial Systems (UAS) in the present world. Able to serve broad application purposes, they are becoming a key technology in many research and industry frontiers. The International Civil Aviation Organization (ICAO) defines a UAS as ”an aircraft and its associated elements which are operated with no pilot on board” [1]. These associated elements include the communication link and the components that control the unmanned aircraft. In the scope of this work, the term UAS will only refer to the vehicle itself.

The absence of onboard human pilots provides them with a set of unique advantages. Not requiring an onboard pilot enables access to challenging and complex scenarios with minimal requirements before take-off, becoming a robust solution for many industries. Modern surveying and mapping industries rely on UAS for their ability to carry sensing devices to inaccessible areas. Their capability to deliver high temporal and spatial resolution information enables a rapid response in critical situations where immediate access to 3D mapping information is crucial [2]. Critical infrastructures need regular inspections to guarantee their integrity. Inspection examples include: bridges and dams, for structural defects and cracks; gas and oil pipelines, for leak detection and equipment corrosion; and electric power grids, for the identification of equipment wear and vegetation encroachment. UAS have become prevalent in such delicate duties as they can reduce costs and minimize human risk by not requiring a hoisted or suspended person while inspecting. Additionally, UAS can sense defects remotely, requiring minimal interaction with the inspected structure [3] [4]. Precision agriculture is a vital sector that currently relies on UAS-based remote sensing systems. More efficient than ground systems, as they can cover a larger field in a short amount of time and in a non-destructive way. UAS can identify which crop areas need to be managed, helping farmers increase productivity and lower costs. [4]. Urban Air Mobility has seen an increasing research interest by companies, academia, and governments. It defines an efficient and safe air traffic operation inside a metropolitan area for UAS. Capable of vertical take-off and landing (VTOL), they present the ideal mobility for urban scenario constraints. Flight operations would include passenger and cargo transport, medical evacuations and traffic management, improving quality of life [5].

Despite all commercial applications, the great majority of UAS belong to the recreational sector. An Aerospace Forecast published by the United States Federal Aviation Administration (FAA) in 2019 [6],

has stated that consumer-grade UAS dominate the commercial sector with a 94% share due to falling equipment prices, improved technologies and built-in sensors, and relatively easy manoeuvring. At the end of 2019, there were 1.32 million small UAS (sUAS) in the United States. The FAA projects their number to rise to over 1.50 million by 2024.

1.1 Motivation

The proliferation of low-price UAS operations constitutes a security and safety threat for both civilian and military organizations. In January 2015, a small quadcopter crash-landed on the White House's south lawn, raising many questions concerning the risk drones pose to public safety [7]. The apparent careless operation highlighted the vulnerabilities of even the most secure infrastructures to an UAS.

Nowadays, a sUAS can jeopardize critical infrastructure and interfere with manned aircraft. Between 19-21 December 2018, Gatwick Airport in London halted its operations due to a drone attack. Authorities claim to be a planned attack from someone with inside knowledge of the airport's operating procedures. The 3-day attack resulted in 140,000 affected passengers and airport losses of around £1.4M [8]. A report from the UK Airprox Board indicates that incidents caused by a sUAS happen regularly in airports around the UK, as seen in Figure 1.1. Even though the current coronavirus pandemic may have reduced the number of reported incidents this year, they may well pick-up again once authorities ease travel restrictions.

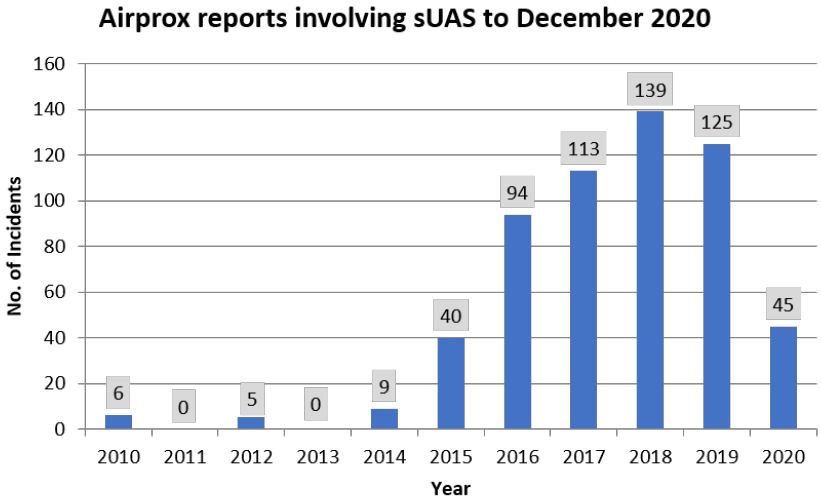


Figure 1.1: UK Airprox Board report on the number of sUAS incidents in UK airports [9].

On the other hand, the pandemic state has increased stealth drones' use to deliver contraband into prison grounds. In Australia, after a ban on face-to-face visits cut common supply routes, 97 incidents related to sUAS smuggling over prison walls have been reported between March and November of 2020 [10].

In civilian airspaces, authorities do not yet require drones to carry transponders, so they cannot be detected and tracked by existing air traffic control systems. sUAS regulations are still under development

in several countries around the world. In Canada, for example, pilots are required to be certified to operate remotely piloted aircraft (RPA) over $250g$ [11]. In Portugal, regulation requires the registration of RPA over $250g$ and a liability insurance for systems over $900g$.

It is possible to modify 'off-the-shelf' UAS into rudimentary guided missiles or other possible airborne attack systems with relative simplicity. Thus, becoming an appealing tool for terrorists and criminals with nefarious intentions. In August 2018, President Nicholas Maduro was attacked by two commercial drones while giving a speech in Caracas, Venezuela, where each RPA contained one kilogram of C-4 explosives [12].

1.2 C-UAS Problem

The development of counter UAS (C-UAS) technologies is related to the recent rise of drone proliferation. Its main objective is to detect, track and sometimes intercept UAS. According to three recent studies from NATO industrial advisory group (NIAG), there is an urgent need to improve existing C-UAS solutions [13–15].

In Table 1.1, NIAG categorizes low, slow and small (LSS) UAS according to their weight, operating altitude, mission range, and payload capacity. Class I vehicles include every UAS with a mass under 150 kg, while Class II extends to larger vehicles between $150kg$ to $200kg$, generally restricted to military aircraft.

The present work will mainly focus on the micro, and mini UAS class, as the top-selling commercially available UAS can be placed in this category [16]. FAA defines sUAS as all unmanned aircraft over 0.55 pounds $(250g)$ and less than 55 pounds $(25kg)$ [6]. In this thesis, the term sUAS will refer to the micro and mini UAS categories presented by NIAG.

Table 1.1: NATO definitions of LSS UAS categories [13].

Class	Category	Operating Altitude (GL)	Mission Radius	Payload
Class I $(< 150KG)$	Micro $(< 2kg)$	To 90 m (300 ft)	5 km	0.2-0.5 kg
Class I $(< 150KG)$	Mini $(2 - 20kg)$	To 900 m (3000 ft)	25 km	0.5-10 kg
Class I $(< 150KG)$	Small $(< 150kg)$	To 1500 m (5000 ft)	50-100 km	5-50 kg
Class II $(150 - 600kg)$	Tactical	To 3000 m (10000 ft)	200 km	25-200 kg

A C-UAS system is only as effective as its capability to detect possible threats. The first line of defence against a threatening sUAS is its early detection and identification. The small size and minimal detection structures of the particular sUAS class makes them hard to detect. The C-sUAS problem is an elaborate multi-stage operation that requires coordination between systems and human operators. It can be divided into four main stages [17]:

1. **Detect, identify, locate, and track**: It defines the task of acquiring all the information needed to understand the threat the target might pose. This information is necessary to make an educated decision on how to respond. Non-cooperative sensors capable of providing this information are described in Section 1.3.

2. **Decision**: It defines the human operator's decision on how to respond to a threat based on sensor information. A human agent needs to evaluate the situation as target mitigation is often a last resort measure.

3. **Mitigation**: It defines the denial of a target's mission, which might include its destruction. Approaches like radio frequency (RF) jamming or GNSS spoofing present a way to intercept a drone by restricting its access to outside signals. Other techniques include the neutralization of a vehicle using nets, collision drones or projectiles.

4. **Retrieval**: It defines the retrieval of a sUAS once the threat has been mitigated. If armed with weapons or explosives, the vehicle may need to be isolated and handled with care. Personal may conduct forensics to examine the danger further.

A study presented in Figure 1.2 estimates that the C-UAS market could value up to 6.6 billion USD by 2024.

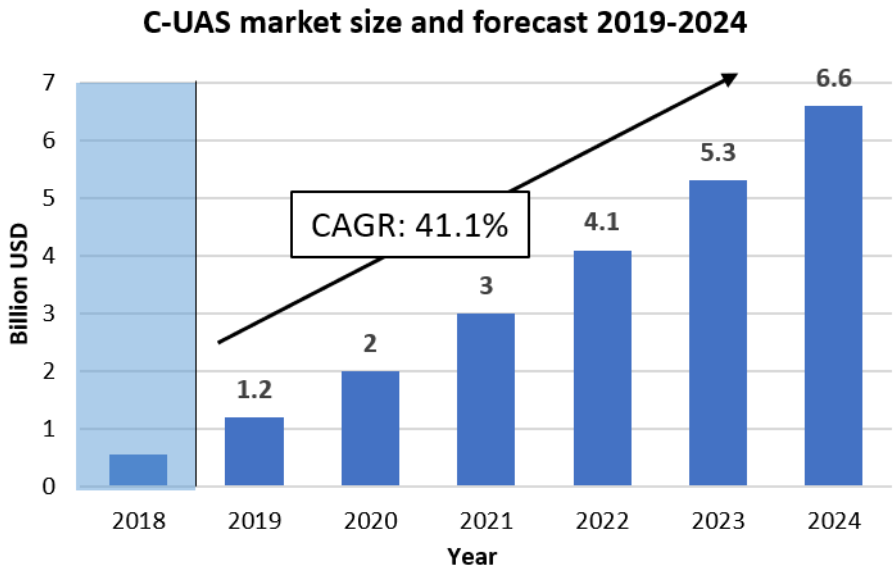


Figure 1.2: Droneii C-UAS market size and forecast 2019-2024, with a compound annual growth rate (CAGR) of 41.1% [18].

1.3 Non-cooperative Sensors

Surveillance methods which do not require the target to transmit information about its position (e.g. transponders) are classified as non-cooperative. Similarly, UAS which lack total or partial broadcast of its position, and which will not actively cooperate to resolve a conflict are classified as non-cooperative.

Non-cooperative sensors are essential for C-UAS applications. They can either be classified as passive or active sensors, depending on whether they transmit the energy needed for object detection. Passive solutions include:

- **Acoustic**: It presents a traditional way to acquire the presence and location of a flying vehicle. Propellers and motors are its main sound sources. Microphone arrays have additional benefits over a single device. Through beamforming, arrays can detect the elevation and azimuth of the arriving sound wave. Beamforming also increases signal-to-noise ratio (SNR), generating more precise detections. All in all, acoustic detection and tracking can be completely passive and relatively inexpensive. However, the range and quality of detections are dependent on environmental conditions, like background noise. Additionally, engineers must consider a special installation strategy for an airborne solution to minimize the aircraft's disturbances.

- **EO/IR Cameras**: The basic principles of electro-optical (EO) and infrared (IR) cameras are the same. They differ on the wavelength of the radiation each captures. While EO cameras perform detections based on the target's visual appearance, IR sensors distinguish the target based on its heat signature. When combined, they can provide situational awareness, both day and night. EO/IR sensors are lightweight and can provide a high-resolution perception of the environment. IR sensors depend on the heat signature of the object to perform a successful detection. Electric powered sUAS produce fewer heat signals than fuel combustion-powered aircraft, the first being the preferred power method for sUAS [19]. The primary heat source for electric-powered sUAS is its batteries, as their large size makes them easier to detect. [20]. However, external factors such as weather conditions for EO cameras, or background temperatures for IR cameras, influence their detection capabilities.

- **Radio Frequency (RF) Sensor**: It can detect, locate, and identify nearby targets by scanning the radio frequencies on which most sUAS operate. RF sensors are relatively inexpensive, and most commercial sUAS emit easily detectable signals. However, they cannot act on a non-transmitting threat.

  Furthermore, active sensing solutions adopted for C-UAS applications include:

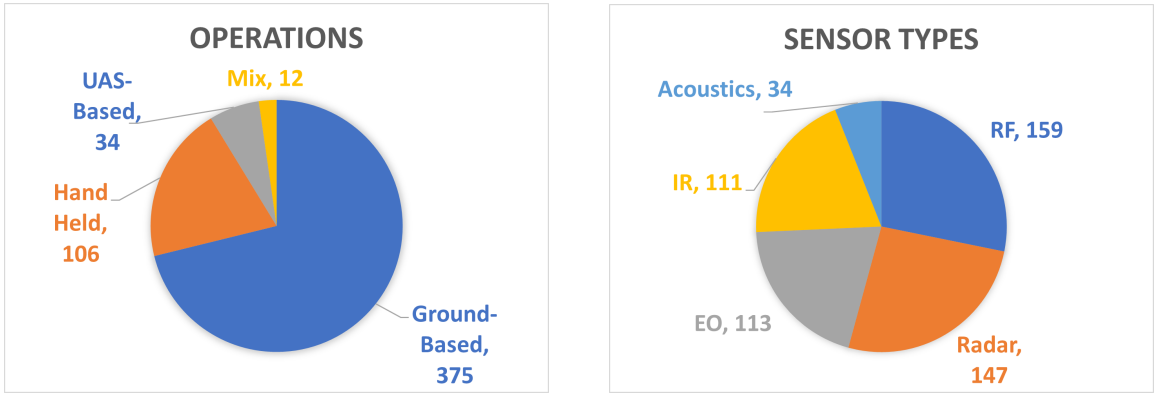
- **Radar**: It identifies a target based on its radar signature, generated when the aircraft encounters RF pulses emitted by the detection element. Radars can operate day and night. However, they present problems detecting small targets, which have low radar cross-sections (RCS) and fly at lower altitudes and speeds than larger aircraft. Solutions to capture small targets include millimetric-wave (mmWave) radars. They enable a precise analysis of the captured sUAS, which might include its classification, the ability to understand if the sUAS is carrying any payload and distinguishing it from other flying objects or animals [21].

- **LiDAR**: It identifies a target similarly to radar, except that the emitted pulse wavelength is in the visible or infrared spectrum. LiDAR provides a 3D representation of the environment and can achieve high update-rates ($5 - 20Hz$). These sensors also can detect vehicles against a complex background and achieve ranges over $100m$. However, this technology is heavy and can not accurately identify small targets due to the created sparse point cloud.

A NATO report [13] on LSS UAS engagement concluded that there is no single sensor type capable

of providing a tracking and identification solution for reliable and effective defence against this threat. So, having the right mixture of sensing technologies is crucial. In [22], the term *sensor fusion*, also known as *multisensor data fusion*, is defined as "the technology concerned with the combination of how to combine data from multiple (and possible diverse) sensors in order to make inferences about a physical event, activity, or situation". The present work shares the same definition.

A market study on available C-UAS solutions is presented in Figure 1.3, evaluating the popularity of operations and sensor types among them. Regarding operational platforms, the vast majority of C-UAS solutions available are ground-based, with 375 systems, while UAS-based platforms only present 34 systems. Additionally, sensing devices such as RF, radar and EO, IR cameras are evenly distributed, with the exception of acoustics, which presents the lowest popularity amongst C-UAS systems. The study also reveals that the LiDAR sensor has not yet breached into the commercial sector, showing there is much to develop in that regard.



(a) Solutions distributed across operational platforms.

(b) Solutions distributed across sensing devices.

Figure 1.3: Market study on available C-UAS solutions [23].

1.4 Related Works

Sensor fusion between a LiDAR and a camera can be seen as an intuitive solution since they both can complement each other's limitations. Most notably, their application for C-sUAS purposes was researched by Hammer et al. [24]. A sensor fusion solution between several LiDAR sensors and a pan-tilt camera was demonstrated on a ground-based vehicle. Initial research [25] showed the capability of multiple high-resolution LiDAR sensors to reliably detect sUAS, presenting a recall of 70% for ranges below 35 m. In [24], the objective was for a pan-tilt camera to align itself towards a flying object previously detected by the LiDAR and use the Faster R-CNN algorithm [26] for its classification. However, the presented solution is based on heavy sensor equipment (i.e. multiple LiDAR sensors and a large pan-tilt unit), unfeasible for an aerial detection solution, which this research aims to achieve.

In the present work, LiDAR detections create search windows inside the image frame. A similar concept is proposed by Opromolla et al. [27] to visually detect and track UAS while airborne. It created a search window by knowing the GNSS position of a cooperative UAS and projecting its predicted location into the image frame. However, large uncertainties in the process led to the formation of an extensive

search region to compensate, as seen in Figure 1.4. Similar to the presented work, search regions are analyzed by a YOLOv2 detector, and tracking by detection solution is adopted. Nevertheless, the cooperative assumption made in [27] is not suitable for a C-UAS solution.

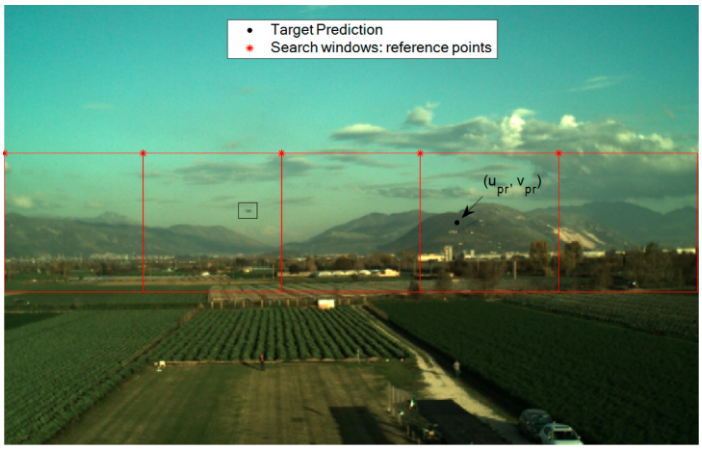


Figure 1.4: Example of the search window, divided into five regions, to be processed by the YOLOv2 detector. The target UAS is represented by a black bounding box and its prediction on the black dot [27].

A single-camera approach to UAS detection and tracking is proposed by Li et al. [28] [29]. It applied background subtraction and optical flow methods onboard an aircraft. Moving objects with respect to the global motion were identified, and salient points in the regions were clustered together. Additionally, a deep learning classifier was applied on patches around each cluster to distinguish between true moving objects from false alarms. The authors recognize that the image subtraction approach can be computationally expensive when applied to high-resolution images. However, no comment was made on the solution's processing speed, so its real-time capabilities are unknown.

Localization of non-cooperative sUAS onboard an aircraft has been attempted by Vrba et al. [30][31] using a combination of a stereo camera and a YOLO-based detection method. The proposed approach reports reliable object tracking up to 32 m, showing that combining both methods improves overall results. Nevertheless, these techniques are not able to directly measure target distances.

Other approaches have researched the usage of onboard active sensors for sUAS detection. Notably, Dogru and Marques [21] demonstrated that an airborne millimetre wave RADAR could measure a drone's bearing and range up to 25 meters. de Haag et al. [32] performed a study on the capabilities of an airborne LiDAR to detect drones weighing less than 250 g, reporting a high detection probability for ranges smaller than 15 m, but leaving heavier vehicles still out of its scope. The main limitation of both these approaches is the inability to identify the flying vehicle.

This thesis aims to fill the literature gap for an aerial sensor system that employs a LiDAR and an EO sensor to achieve real-time detection and tracking of non-cooperative sUAS.

1.5 Objectives

This thesis aims to develop a sensor system capable of detecting and tracking sUAS. The system should make use of non-cooperative sensor, as to be compatible with a C-sUAS solution.

The objective is to develop a sensor fusion solution between a LiDAR and a camera to visually detect and track multiple sUAS in real-time. The sensor fusion methodologies should be compared to single sensor techniques to evaluate the overall benefits. The algorithm should have real-time capabilities.

The proposed approach aims to use LiDAR point clouds to acquire target locations and then to use a visual system to detect and track each vehicle. The LiDAR should alleviate the visual detector's search task by providing it with regions of interest (ROI) around each target. Once acquired, the visual system should be able to track the vehicle on its own.

Experiments should be performed with a sensor system onboard an aircraft while capturing multiple sUAS targets. The goal is to use the collected data to evaluate the proposed methodologies.

1.6 Contributions

With this work, the following contributions are made:

- Development of a multi-sensor methodology for sUAS detection and tracking;
- Demonstration of an extrinsic calibration procedure between a LiDAR and an EO camera;
- Construction and evaluation of a sensor system onboard an aircraft;
- Creation of a labelled dataset for drone detection.

The research's preliminary findings were presented in a paper titled "LiDAR and camera sensor fusion for air-to-air detection and tracking of airborne intruders", submitted to the Unmanned Systems Canada conference on November 3^{rd} 2020, being distinguished as one of the finalists in the "Student Paper Competition" [33].

1.7 Thesis Outline

This thesis is structured as follows: Chapter 2 provides background knowledge for the present work. Chapter 3 explains the methods related to the sensor fusion solution and the calibration applied to the LiDAR and camera sensors. Chapter 4 describes the assembly of the payload system and the flight test experiments conducted. Chapter 5 presents and discusses the results obtained. Lastly, Chapter 6 summarizes the overall findings of this thesis and outlines possible steps for future work.

Chapter 2

Background

This chapter discusses the background related to the presented work. Initially, it approaches the visual detection and tracking tasks (Sections 2.1 and 2.2). It then makes an overview on the LiDAR working principles (Section 2.3), followed by the theory behind intrinsic and extrinsic camera calibration (Section 2.4).

2.1 Visual Object Detection

The goal of visual detection is to enclose an object within a bounding box and determine what object it is. This section provides a historical overview of object detection (Section 2.1.1) followed by an explanation of the YOLO working principles (Section 2.1.2) and subsequent improvements (Section 2.1.3).

2.1.1 Historical Overview

Real-time detection of human faces was achieved for the first time by P. Viola, and M. Jones [34] in 2001. The algorithm, known as the Viola-Jones detector, used the most straight forward way for object detection: sliding windows, i.e. go through all possible locations and scales in an image to see if any window contains a human face. Although a simple process, the necessary calculations surpassed the computational power of its time. Nevertheless, the Viola-Jones detector was able to achieve a drastic increase in speed performance due to three crucial techniques. The first introduced a new image representation called 'Integral Image', allowing the object's features to be computed exceptionally fast. The second is a learning algorithm based on AdaBoost, which selects a small number of critical information from visual features efficiently. The last component increasingly combined more complex classifiers in a 'cascade', quickly removing background regions of the image while spending more time on essential areas, such as human faces.

Histogram of Oriented Gradients (HOG) feature descriptor, initially proposed by N. Dalal and B. Triggs [35] in 2005, is considered an important milestone in the scale-invariant feature transforms of its time. The HOG descriptor focuses on analyzing the object's structure or shape by extracting the direction of image pixels gradients. Gradients with large magnitudes (regions of abrupt intensity changes) typically

indicate edges and corners, some of the most relevant features in an object. An image is split into different regions to be analyzed separately. Finally, the HOG descriptor creates a histogram for each region using the pixel values' gradients and orientation. Mainly suited for human detection, HOG has been an important foundation for many object detectors.

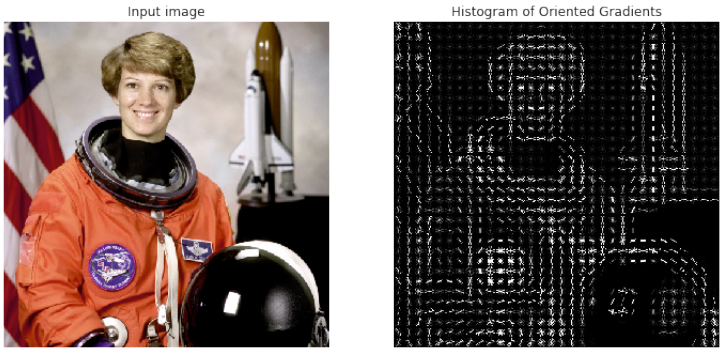


Figure 2.1: Visualization of the HOG features representative of an image of the astronaut Eileen Collins [36].

The Deformable Part-based Model (DPM), proposed by Felzenszwalb et al. [37] in 2008, is considered the peak of traditional object detection methods. Based on the HOG detector, the DPM is motivated by the lack of ability to handle geometric deformations. Follows the detection philosophy of "divide and conquer" by teaching the training task how to decompose an object correctly. For example, the task of detecting a car can be decomposed into detecting its wheels, windows, and body. A latent support-vector machine (SVM) learns how to identify the object based on relationships between HOG features extracted from its components.

In 2012, Krizhevsky et al. won the ImageNet competition with AlexNet [38]. The concept was not entirely new, as the first multi-layered CNN named convNET has already been proposed by Y. LeCun et al. in 1989 [39], rooted in Fukushima's Neocognition [40]. However, it demonstrated the ability of deep architectures to improve the representative capacity of a CNN. Thus, it marked the beginning of the deep learning era, where object detection started to evolve at an unprecedented speed. Now, approaches to object detection generally fit into two main categories: two-stage and single-stage detectors.

Two-stage methods perform the detection task by first generating regions of proposal, instead of sliding windows, followed by their classification. Generally, two-stage object detectors are accurate but computationally expensive. Examples remote back to 2014 when R. Girshick proposed Regions with CNN features (R-CNN) for object detection [41]. In its essence, the algorithm begins by creating regions proposals via a selective search [42]. From there, features are extracted via a CNN and classified by a SVM. A subsequent improvement is proposed in Fast R-CNN [43], which uses a CNN for both feature extraction and classification. Finally, Faster R-CNN [26] introduces a novel Region Proposal Network (RPN), excluding the selective search. This way, the algorithm applied a CNN end-to-end.

Single-stage methods perform both the localization and classification of the object in a single step. They present fast processing speeds with moderate detection accuracy. A popular single-stage solution has been the YOLO detection algorithm [44]. The following section explains its basic operating principles.

2.1.2 YOLO Overview

The detection process of the YOLO algorithm is represented in Figure 2.2. Given an image, YOLO begins by splitting it into an $S \times S$ grid. Each cell is responsible for predicting B bounding boxes consisting of five variables: x, y, w, h and confidence. The coordinates (x, y) represent the box centre, (w, h) the width and height of the box, and the confidence score predicts the intersection over union (IOU) between boxes B and any ground truth. Each bounding box is ranked based on its confidence score. However, this step does not yet determine the object, just the confidence of any object present in that region. Each cell predicts C conditional class probabilities, and regardless of the number of boxes B associated with that cell, only one class per cell is selected. Being a conditional probability, it does not, for example, directly state the probability of a bicycle, but rather the probability of it being a bicycle knowing there is an object. Each B bounding box's confidence score is multiplied by the single conditional class probability specific to that cell, giving a class-specific confidence score. The final prediction results in a $S \times S \times (B \times 5 + C)$ tensor.

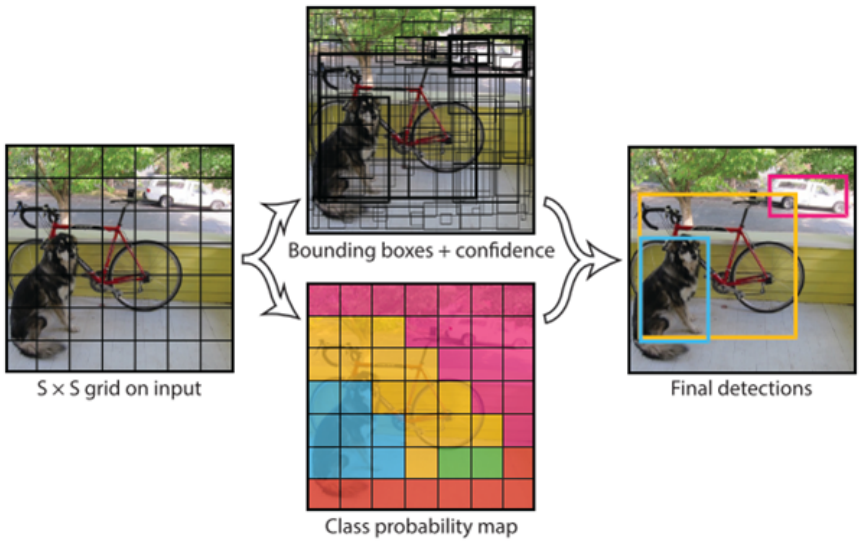


Figure 2.2: Detection process of the YOLO algorithm. The left image represents the $S \times S$ grid cells. The top image shows the B bounding boxes. The bottom image, representing the conditional class probability for each cell. Finally, the right picture presents the final predictions [44].

As many bounding boxes have low prediction scores, the algorithm will keep only the highest ones by imposing a confidence threshold. Additionally, as multiple bounding boxes can be related to the same object, a non-maximum-suppression algorithm is applied. It relies on a confidence and IoU threshold to remove overlapping boxes, keeping the ones with the highest prediction score. By predicting all objects in the image simultaneously, the YOLO algorithm can incorporate global context in the detection process, requiring only one network evaluation. Hence its name of "You Only Look Once".

2.1.3 YOLO Improvements

Various improvements to the algorithm are made in its second version, called YOLOv2 [45]. Batch normalization is added to the training process, making the learning process much faster. Additionally, dimension clusters are included in the algorithm. Like Faster R-CNN, it employs anchor boxes, but instead of using predefined and fixed dimensions, the algorithm tunes anchor boxes based on the ground truth targets. Thus, it can detect a greater variety of objects with fewer detection boxes, resulting in a faster and more accurate solution. Finally, the algorithm introduces multi-scale training, where images in the training dataset are scaled to resolutions multiple of 32 during the training process. This technique forces the same network to predict correctly across different dimensions. This second version improved the mean average precision (mAP) of the original algorithm, although achieving a lower framerate, as seen in Table 2.1.

Table 2.1: Results obtained on PASCAL VOC 2012 dataset [46].

Method	mAP	Runtime
YOLO [44]	57.9	45 Hz
YOLOv2 [45]	78.2	40 Hz

In its third version, YOLOv3 [47] introduced new improvements on top of the previous algorithm. In it, the ability to deal with small targets was drastically increased. Its feature extraction is inspired by feature pyramid networks [48], predicting object locations across three different scales. The concept uses backward connections in the convolutional network to merge information between later and earlier layers. It helps predict detections at higher resolutions, as later layers in the network have strong semantic information and earlier layers provide high-resolution features. Compared to the previous version, YOLOv3 achieved higher mAP, at the cost of a lower processing speed, as shown in Table 2.2.

Table 2.2: Results obtained on COCO2017 dataset [49].

Method	mAP	Runtime
YOLOv2 [45]	48.1	40 Hz
YOLOv3 [47]	55.3	35 Hz

2.2 Visual Tracking

2.2.1 Overview of Tracking Task

Tracking is the problem of generating an inference about an object's motion, given a sequence of images. A moving object must have a state to be tracked, that being its position, velocity, or appearance. Each measurement is called an observation. In many problems, observations are functions of the state, with some added noise. For example, the state might be the position and velocity, and only the position is observed. In tracking problems, there is a model for how the state changes with time, referred to as the object's dynamics. Tracking involves exploiting both observations and dynamics to infer the object state. The most critical visual tracking property is that observations are usually hidden in a great deal

of irrelevant information. There are two main methods to identify which observations are likely to be helpful: tracking by detection and tracking by matching.

In tracking by matching, there is a model for how the object moves and appears, and given a domain in the n^{th} frame in which the object is located, the model is used to search for a domain in the $n + 1^{th}$ frame that matches it. It works well when the target has not abruptly changed its appearance or the scenario too complicated. The object's appearance model is generated on previous frames, and objects in the current frame with the closest distance to the samples are selected.

In tracking by detection, there is a strong model of the object to identify it in each frame. By linking each detection, it is possible to construct a track. The tracking by detection technique is quite general and extremely effective. However, the association step's main issue is the cost model, which will change depending on the application. For slow-moving objects, the cost can be the image distance between the detection on the current frame and the previous. For objects with a slowly changing appearance, the cost can be an appearance distance. The association problem can be formulated as a bipartite matching problem and solved with the Hungarian algorithm [50].

2.2.2 DeepSORT Algorithm

Simple Online and Real-time Tracking (SORT) was published by A. Bewley et al. [51] in 2016. It is a simple and pragmatic tracking algorithm based on the tracking by detection formula. The cost function of the bipartite matching problem consists of an intersection over union (IOU) metric, thresholded to reject non-ideal detections. This association metric helps to deal with occlusions by giving preference to bounding boxes of similar ratio and size when two targets pass in front of each other. The problem is solved optimally using the Hungarian algorithm.

SORT with a deep associative metric (DeepSORT) was published by N. Wojke et al. [52] in 2017, and presents an extension to the SORT tracking solution. It assumes the camera is uncalibrated and there is no ego-information available. For the tracking task, it models the object state x as

$$x = [u, v, \gamma, L, \dot{x}, \dot{y}, \dot{\gamma}, \dot{L}]^T \tag{2.1}$$

where (u, v) is the bounding box centre, γ is the aspect ratio, L is the height, followed by the respective velocities. State observations correspond to bounding boxes that provide $[u, v, \gamma, L]$.

Each time a target i has an observation assigned, its state x_i is updated, with the velocity components being solved by a Kalman filter. In frames where no target observations are associated with a given target i, x_i is updated using a linear velocity model.

The bipartite matching problem is constructed with a cost function comprised of a weighted sum of two different metrics: distance-based and appearance-based.

For the distance metric, it computes the Mahalanobis distance between predicted Kalman states and newly arrived observations. It takes into account how many standard deviations the measurement is away from the mean track location. The Mahalanobis distance has a threshold t_M at a 95% confidence interval, computed from the $\mathcal{X}^2$ distribution. For a four dimentional measurement space this value is

$t_M = 9.4877$.

The appearance metric is computed via a cosine distance between feature descriptors. First, a CNN extracts appearance descriptors from the obtained bounding box. It then computes the smallest cosine distance between it and a list of previously associated descriptors.

If an observation is not associated with any of the tracks, a second matching cascade problem is constructed with the IOU metric from the original SORT algorithm.

2.3 LiDAR Fundamentals

Light detection and ranging (LiDAR) sensors work equivalently to radars, but rather than microwave energy emitted by an antenna, they exploit the light emitted by a laser beam. A laser is an optical device that, when activated by an external energy source, produces and emits a beam or pulse of radiation with the following attributes:

- **Coherent:** all photos emitted at the same time, have the same phase.
- **Monochromatic:** all photons have almost the same wavelength or frequency.
- **Collimated:** photons emitted move on parallel rays, rather than on large beams, as happens on radars.

Operating laser systems in human environments can be dangerous, as laser irradiation on the eye may cause damage to the cornea lens or retina. Standards distinguish laser with classes depending on their potential to cause harm. Class-1 laser systems are considered to be incapable of producing damaging radiation levels. The combination of low power, rapid laser rotation and careful laser wavelength placement (usually, $\lambda = 905nm$) guarantees the LiDAR sensor is not harmful to human eyes.

LiDAR measurements are a function of both sensor and target characteristics. This relationship is described by the standard LiDAR equation [53], derived from the traditional radar equation, which relates the power of transmitted (P_t) and received (P_r) signals:

$$P_r(t) = \frac{D_2}{4\pi\lambda^2} \int_0^H \frac{\eta_{sys}\eta_{atm}}{R^4} P_t \left(t - \frac{2R}{v_g}\right) \sigma(R) dR \tag{2.2}$$

where t is time, D the aperture diameter of the receiver optics, λ the laser wavelength, H the distance travelled by the laser, R the distance from the system to the target, η_{sys} and η_{atm} are respectively the system and atmospheric transmission factors, v_g the group velocity of the laser pulse and $\sigma(R)dR$ the apparent effective differential cross-section. The term ”apparent” is used since an object reflecting the signal at a given distance can occlude an object further away. The received power $P_t(t)$ can be seen as a contribution of N targets, and written as:

$$P_r(t) = \sum_{i=1}^{N} P_{r,i}(t) \cdot \eta_{sys}(t) \cdot \eta_{atm}(t) = \sum_{i=1}^{N} \frac{D_2}{4\pi\lambda^2 R_i^4} P_t(t) \cdot \eta_{sys}(t) \cdot \eta_{atm}(t) \cdot \sigma_i^{'}(t) \tag{2.3}$$

where $\sigma_i^{'}(t)$ is the apparent cross section of illuminated areas within each range interval, $P_t(t) \cdot \eta_{sys}(t)$ the component from the system contribution and $\eta_{atm}(t) \cdot \sigma_i'(t)$ is the environmental contribution.

Although not unique, LiDAR sensors usually compute distances by measuring the time of flight (TOF) of a very short but intense pulse of laser radiation. This simple calculation can be represented as

$$R = \frac{c \cdot (t_r - t_t)}{2} \tag{2.4}$$

where R is the distance between the ranging unit and the object surface, c is the speed of electromagnetic radiation, and t_r and t_t are respectively the received and transmitted time measurements. Range measurement R can be transformed into the sensor's frame of reference with

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} cos(\alpha) \cdot cos(\omega) \\ cos(\alpha) \cdot sin(\omega) \\ sin(\alpha) \end{bmatrix} \tag{2.5}$$

where α is the elevation, and ω the azimuth angle of the emitted laser beam [54].

2.4 Projective Geometry

This section starts by presenting the intrinsic camera parameters: the camera calibration matrix, and the distortion coefficients caused by the camera lens (Section 2.4.1). It then describes the camera pose estimation problem (Section 2.4.2).

2.4.1 Intrinsic Camera Parameters

The pinhole perspective camera model (also known as the central perspective) considers that all the light from a scene that reaches the optical sensor must pass through an ideal pinhole O. Thus, exactly one light ray would pass through each point in the image plane.

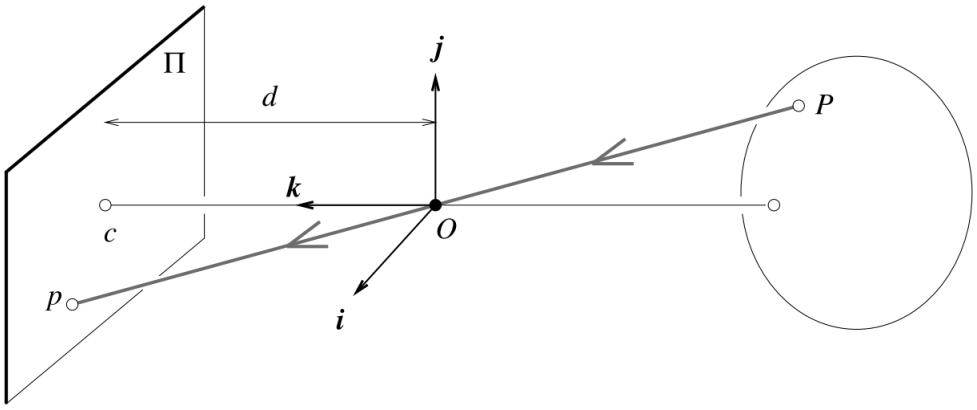


Figure 2.3: Representation of the collinearity between point P, its image p and the pinhole O [50].

Let P denote a scene point with coordinates (X, Y, Z), and p its image coordinate (x_c, y_c, d). From

Figure 3.4, it follows that the three points P, O and p are collinear, meaning that for some number λ, $\vec{Op} = \lambda\vec{OP}$, so:

$$\begin{cases} x = \lambda X \\ y = \lambda Y \\ d = \lambda Z \end{cases} \Leftrightarrow \lambda = \frac{x}{X} = \frac{y}{Y} = \frac{d}{Z} \tag{2.6}$$

therefore,

$$\begin{cases} x = d\frac{X}{Z} \\ y = d\frac{Y}{Z} \end{cases} \tag{2.7}$$

In the camera model, points are projected onto the image plane by dividing them by their z component. This implies that it is not possible to recover distances from the image plane, which makes sense for a 2D projection.

Assuming the camera is focused on infinity, it means the plane distance is equal to the focal length $d = f$. Let the point $\hat{p} = (\hat{x}, \hat{y}, 1)^T$ be the vector of homogeneous coordinates of p when transformed into the normalized image plane, located at an unit distance $d = 1$ from the pinhole O. Then,

$$\hat{p} = \frac{1}{Z}\begin{bmatrix} Id & 0 \end{bmatrix}\hat{P} \tag{2.8}$$

where $\hat{P}$ is the vector of homogeneous coordinates of P. However, the real camera plane may not actually coincide with the normalized image plane, having a distance $d \neq 1$. Additionally, its pixel coordinates may have a different centre $c = (c_x, c_y)$, and its pixels rectangular instead of square. The conversion from point $\hat{p}$ into point p is obtained through the camera calibration matrix K:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K\hat{p} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} \tag{2.9}$$

where f_x and f_y are the focal lengths expressed in pixels, s represents any skew between sensor axes due non-rectangular pixels, and $(\hat{x}, \hat{y})$ the optical centre expressed in pixel coordinates [50] .

Until this point, the camera model discussed assumes a linear projection. This means straight lines in the 3D world are preserved inside the image. However, many camera lenses have noticeable radial distortions, which manifest as projected curved lines that otherwise would be straight. Compensating for this effect is not difficult. Let us assume $(\hat{x}, \hat{y})$ is the pixel coordinates obtained after the central perspective projection but before the scaling and shifting effects of the K matrix. The radian distortion model states that image pixels are displaced either towards or away from the image center as a function of their distance to the image centre. Thus, radial distortions can be modelled as a polynomial function:

$$\begin{cases} \hat{x}_r = \hat{x}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \\ \hat{y}_r = \hat{y}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \end{cases} \tag{2.10}$$

where $(\hat{x}_r, \hat{y}_r)$ are the coordinates of the point affected by radial distortion, $r^2 = \hat{x}^2 + \hat{y}^2$ and k_1, k_2, k_3 are the radial distortion parameters.

Tangential distortion occurs when the image plane and the camera lens are not parallel with each other. It can me modelled as:

$$\begin{cases} \hat{x}_t = \hat{x} + (2 \cdot p_1 \cdot \hat{x} \cdot \hat{y} + p_2 \cdot (r^2 + 2\hat{x}^2)) \\ \hat{y}_t = \hat{y} + (p_1 \cdot (r^2 + 2\hat{y}^2) + 2 \cdot p_1 \cdot \hat{x} \cdot \hat{y}) \end{cases} \tag{2.11}$$

where $(\hat{x}_t, \hat{y}_t)$ are the coordinates of the point affected by tangential distortion, and p_1, p_2 are the tangential distortion parameters [55].

2.4.2 Camera Pose Estimation

To project 3D points from the world frame of reference into the camera plane, one must know the camera pose relative to it. This projection may be modelled as

$$p = \mathcal{M}\hat{P} \tag{2.12}$$

were $p = (x, y, 1)^T$ are the image points in homogeneous coordinates, $\hat{P} = (X, Y, Z, 1)^T$ are the world points, and $\mathcal{M}$ is the 3×4 camera projection matrix. The matrix $\mathcal{M}$ can be decomposed as

$$\mathcal{M} = K \begin{bmatrix} R & | & t \end{bmatrix} \tag{2.13}$$

where K is the calibration matrix explained in Section 2.4.1, and the rotation R and translation t matrices represent the Euclidean transformation between the camera and the world coordinate system, i.e. the camera pose. In total, the camera pose has six degrees-of-freedom, three in the translation matrix t and three on the rotation R.

The problem of finding the camera pose given its calibration matrix K and a set of N 3D-2D point correspondences is known as perspective-n-point (PnP). One solution to the PnP problem is the Efficient PnP (EPnP) algorithm, proposed by V. Lepetit et al. [56], which solves the problem in $\mathcal{O}(n)$ time. It assumes the input points are non-collinear and outliner-free. However, this is not the case for most applications, as the set of point correspondences provided to EPnP have associated errors. Thus, EPnP usually serves as an initial estimation that is then optimized by an iterative method, such as the random sample consensus (RANSAC) [57], leading to a more robust solution.

2.5 Performance Metrics

This section presents the performance metrics used to evaluate the approach taken in this thesis.

2.5.1 Detection Metrics

This subsection presents the performance metrics related to object detection. They are as follows:

•**Intersection Over Union**:

In the visual detection task, the algorithm is expected to localize an object in the image. To evaluate if the predicted bounding boxes are well adjusted to the objects the algorithm measures the intersection over union, IOU:

$$IOU = \frac{Area(B_P \cap B_{GT})}{Area(B_P \cup B_{GT})} \tag{2.14}$$

where B_P is the predicted bounding box and B_{GT} is the ground truth bounding box. Figure 2.4 presents a visual representation of this metric. If no overlap is presented between bounding boxes, $IOU = 0\%$. In contrast, if both bounding boxes overlap entirely, $IOU = 100\%$. Two bounding boxes match if $IOU \geq 50\%$, as set in the PASCAL VOC competition [58].

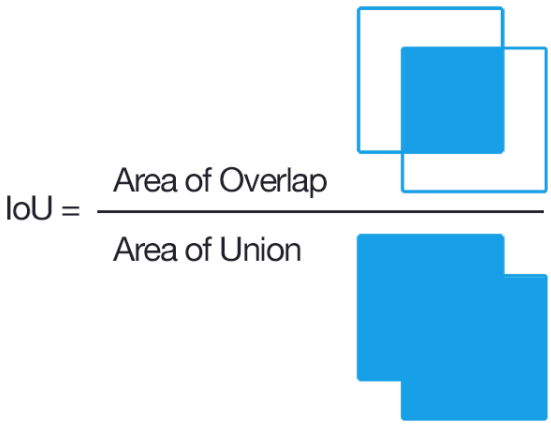


Figure 2.4: Visual representation of the intersection over union (IOU) metric.

•**Precision**:

Intuitively, the precision, P, is the ability of an algorithm to perform correct predictions. It is defined as

$$P = \frac{TP}{TP + FP} = \frac{TP}{total\ detections} \tag{2.15}$$

where TP is the number of true positive and FP is the number of false positive. A detection is counted as a TP if $IOU \geq 50\%$. Otherwise, it is viewed as a FP. The precision values, in percentage, range between $[0, 100]$, where $P = 100\%$ corresponds to an algorithm which does not present any false positives, assuming $(TP + FP \neq 0)$.

•**Recall**:

Intuitively, the recall, R, is the ability of an algorithm to find all the available ground truths. It is defined as

$$R = \frac{TP}{TP + FN} = \frac{TP}{total\ objects} \tag{2.16}$$

where FN is the number of false negatives in the set and it represents a ground truth label that is not detected. The recall values, in percentage, range between $[0, 100]$, where $R = 100\%$ corresponds to an algorithm able to correctly predict every available object, assuming $(TP + FN \neq 0)$.

The LiDAR recall, R, is computed using the same formula. For a LiDAR point cloud, there is no IOU metric. In this scenario, the number of TP and FN are obtained by inspecting if each point's coordinates are coherent with the actual target location.

•**F1-score**:

The F1 score, $F1_{score}$, is defined as the harmonic mean of precision, P and recall, R, and is given as

$$F1_{score} = \frac{2 \cdot P \cdot R}{P + R} \tag{2.17}$$

•**Precision-Recall Curve**:

Predictions made by a visual detector have an associated confidence score, c_{score}, on which they decide ether to discard or to keep it, based on a predefined threshold, c_{thres}. The prediction is kept if $c_{score} \geq c_{thres}$.

Each c_{thres} value provides the visual detector with a different set of P and R metrics. By going through all c_{thres} value possibilities, a precision-recall (P-R) curve can be constructed, as seen in Figure 2.5. For a high c_{thres}, only predictions with the highest confidence scores are kept, providing the detector with high precision but low recall values. Inversely, low c_{thres} values means more predictions are kept, increasing target recall.

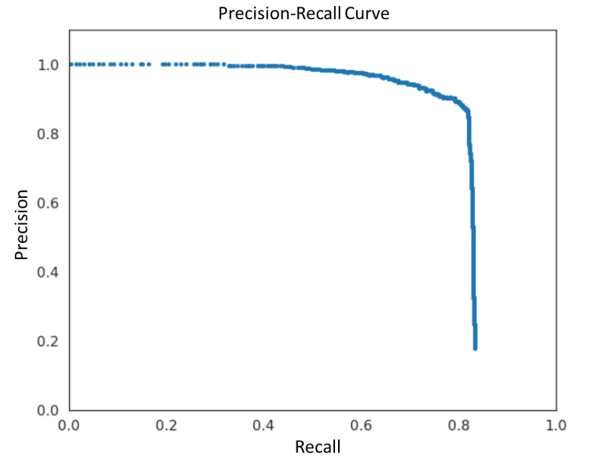


Figure 2.5: Example of a precision-recall curve.

•**Average Precision and Mean Average Precision**:

The average precision, AP, compares the performance of object detectors by calculating the area under the P-R curve. For a P-R curve, AP is defined as

$$AP = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1}) \tag{2.18}$$

where R_n is the recall value of point n and $P_{interp}(R_{n+1})$ is the maximum precision for any recall value $\widetilde{R}$ bigger than R_{n+1}, meaning:

$$P_{interp}(R_{n+1}) = \max_{\widetilde{R}:\widetilde{R} \geq R_{n+1}} P(\widetilde{R}) \tag{2.19}$$

The mean average precision, mAP, is simply the average AP across different classes, given by

$$mAP = \frac{1}{N} \sum_{j=1}^{N} AP_j \tag{2.20}$$

where N is the total number of classes and AP_j is the average precision of the j^{th} class.

2.5.2 Tracking Metrics

This section presents the metrics used to evaluate the algorithm for multiple object tracking (MOT), and their definitions are based on [59]. They are as follows:

•**Fragments**:

The number of fragments, FM, counts how many times a ground truth trajectory is interrupted. It is increased each time a trajectory changes its status from tracked to untracked, and the tracking of the same trajectory is resumed at a later point.

•**Identity Switches**:

A mismatch, or equivalently, an identity switch, $IDSW$, is counted if a ground truth target i is matched to track j and the last known assignment was $i \neq j$.

•**Track Quality Measures**:

Each ground truth trajectory can be classified as mostly tracked, MT, partially tracked, PT, and mostly lost, ML, based on how much of the track is recovered by the tracking algorithm. A track is MT if at least 80% of its length is recovered. If the recovered track is less than 20% of its length, it is classified as ML. In all other cases, the track is classified as PT.

•**Multiple Object Tracking Precision** :

The multiple object tracking precision, $MOTP$, computes the average dissimilarity between all true

positives and their corresponding ground truth targets. It is given by

$$MOTP = \frac{\sum_{t,l} d_{t,l}}{\sum_t c_t} \tag{2.21}$$

where $d_{t,l}$ is the IOU value of target l with its assigned ground truth object in frame t and c_t is the total number of matches in frame t, and ranges, in percentage, between $[50, 100]$.

•**Multiple Object Tracking Accuracy**:

The multiple object tracking accuracy, $MOTA$, a metric used to evaluate the tracker's performance. It combines three sources of errors and is defined as

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{2.22}$$

where t is the frame index, and GT is the number of ground truth objects. The MOTA may be represented as a percentage, with a possible range between $[-\infty, 100]$. Negative cases happen when the number of errors made by the tracker exceeds the number of all objects in the scene.

Chapter 3

Methodology

This chapter presents the methods of the proposed approach. It starts by giving an overview of the sensor fusion solution created (Section 3.1), followed by a summary of the visual detection and tracking algorithm developed, i.e. *YOLO-based tracker* (Section 3.2). It then describes the LiDAR pre-processing steps (Section 3.3) and the calibration procedures (Section 3.4). This chapter finishes by explaining the region of interest (ROI) creation (Section 3.5).

3.1 Sensor Fusion Methodology

As previously mentioned, electro-optical (EO) cameras can capture the environment in high-resolution. Based on such rich information, it is possible to identify and classify targets based on extracted features. When looking for aerial vehicles, targets are most often only small objects within a larger context. For example, a target measuring $45cm \times 45cm$, captured by an EO camera at 50 m, only occupies a 20×8 pixel area inside a 1280×720 image. Actively searching in the entire image is not only computationally expensive but can also lead to the acquisition of a substantial amount of false positives. Thus, refining the area of search would prove extremely valuable.

LiDAR sensors can provide direct target measurements. The point clouds generated are fast to create (i.e. up to $20Hz$) and to process. Furthermore, if employed in an aerial scenario, only a portion of the laser beams emitted will generate returns, as most will point towards the sky. So, there is a substantial likelihood that the observed returns correspond to an aerial target.

Nevertheless, point clouds usually contain non-desired points, which must be filtered. Additionally, since multiple points can represent a single target, a clustering method is also important to identify objects of interest among the cloud.

A sensor fusion methodology between a LiDAR and an EO camera is created to outperform single sensing solutions. Developing a 3D-2D point projection method is fundamental to correlate information between both sensors. This calibration enables LiDAR acquisitions to be projected into the camera frame for further inspection. Also, creating regions of interest (ROI) around projected clusters provides the visual detection algorithm with a refined search area to process.

The use of a visual tracker can establish temporal consistency between visual observations. Additionally, by using an internal estimator, such as a Kalman filter, the visual tracker can predict future target locations and feed that information back into the ROI creation. This enables the creation of a closed tracking loop.

On the approach taken, the YOLOv3 is chosen as the visual detector, complemented by a modified DeepSORT tracker. The combination of both solutions creates the visual detection and tracking system nicknamed *YOLO-based tracker*.

Figure 3.1 presents a simplified diagram of the sensor fusion solution, aiming to detect and track sUAS. The following sections describe each element in the sensor fusion solution in further detail.

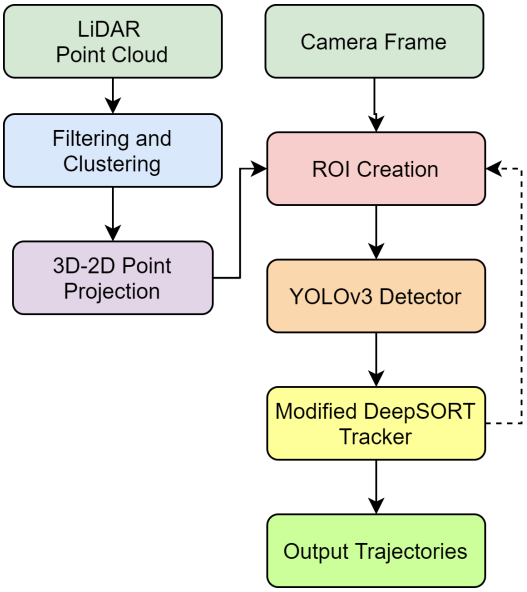


Figure 3.1: Diagram of the sensor fusion algorithm. The dashed line represents a link between consecutive iterations.

3.2 YOLO-Based Tracker

This section approaches the *YOLO-based tracker's* implementation. It is considered to be the backbone of the sensor fusion algorithm. It consists of a YOLOv3 detector and a modified DeepSORT tracker.

3.2.1 YOLOv3 Detector

The visual detection algorithm was chosen based on two factors: accuracy and processing speed. The latter is essential to create a real-time solution. The YOLOv3 algorithm was chosen by presenting a good compromise between both factors.

It was adopted a Pytorch implementation of YOLOv3[1], which provides the source code to train and use the algorithm in a simple way. It also gives a file with pre-trained weights on the MS COCO dataset,

[1] https://github.com/ultralytics/yolov3

on a total of 80 classes, presenting a great starting point to train the neural network on a customized dataset. Even if the dataset contains only a few thousand images, the training process can show great results. This is due to a process known as transfer learning: the idea of using knowledge already acquired in a previous task to solve a similar problem.

The necessity of using a custom dataset comes from the lack of ability from the original MS COCO dataset to accurately detect sUAS. YOLOv3 is trained on a dataset made available by Svanström [60], which is comprised of IR, EO and acoustic data capturing various object classes, like airplanes, drones, birds and helicopters. As this work is focused on sUAS, only the relevant sections of the dataset are extracted and used in the training process. In particular, the portion of the dataset corresponding to EO camera data capturing drones, which includes 114 videos with a 640×512 resolution, totalling $15,133$ labelled image frames. Figure 3.2 presents some image samples from this dataset. The videos were randomly divided into training and validation sets in an $80/20$ proportion. Once the detection performance on the validation set started to degrade, the training procedure was stopped to prevent over-fitting. The training process finished after 20 epochs with $AP = 87.3\%$ and $F1_{score} = 83.1\%$.

Figure 3.2: Sample images from the drone dataset used to train the YOLOv3 algorithm [60].

3.2.2 Modified DeepSORT Tracker

The proposed tracking solution is based on the DeepSORT algorithm explained in Section 2.2.2. The DeepSORT algorithm was initially designed to track pedestrians, which have large and distinct features. It was also intended to be used in high-density scenarios where occlusions are problematic for the distance metric.

Some key modifications are made to adapt DeepSORT for sUAS tracking. Specifically, the CNN appearance descriptors are discarded, as they are not particularly useful when dealing with sUAS. Their small features against complex backgrounds are very difficult to compare. Additionally, the proposed scenario deals with a small number of sparce flying targets, making the distance metric ideal for the assignment problem.

The state machine for a given track is presented in Figure 3.3. For each new and non-assigned observation, a new track is created and placed into a 'tentative' state. After three consecutive associations, the track will transition into an 'active' state ($a1$). If it fails to associate a single one, it transitions into an 'inactive' state ($a2$) to be removed. This requirement for temporal consistency protects the algorithm from sporadic false positives.

The track will remain in an 'active' state until no observation is assigned to it ($a3$). If no associations are made, the track transitions into a 'lost' state ($a4$). While in this state, the Kalman filter inside the DeepSORT tracker will continue to predict the target position based on previous observations ($a5$). If successful in associating with an observation, the track will transition back to an 'active' state ($a6$); however, if no association is made for N iterations, the track transitions into the 'inactive' state ($a7$).

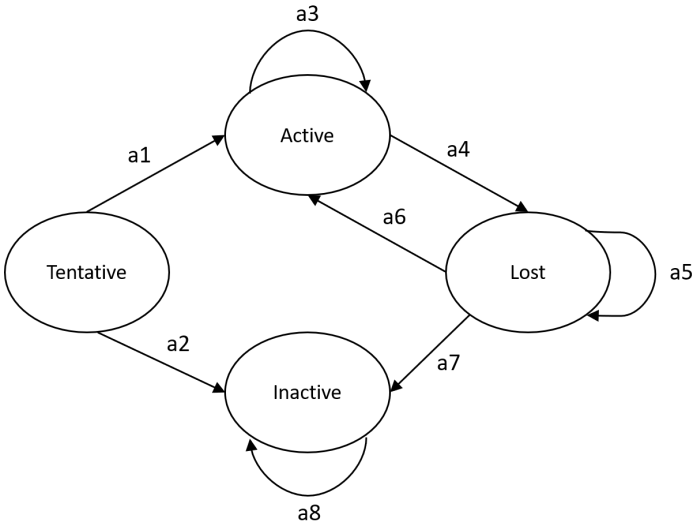


Figure 3.3: State machine for each track in the modified DeepSORT tracking algorithm.

3.3 Point Cloud Pre-Processing Methods

This section presents the two pre-processing steps applied to the point cloud to obtain objects of interest from the LiDAR: filtering and clustering.

3.3.1 Point Cloud Filtering

The filtering step on the point cloud is critical to obtain good LiDAR detection results. The objective is to remove points not of interest for the target detection task. Such points could be ground returns or environment clutter (e.g. trees, buildings). The pseudo-code for the filtering methods is presented in Algorithm 1.

The algorithm assumes that airspace boundaries representing the clutter-free environment are given relative to the sensor's X and Y position. Any point found inside this region corresponds to a possible target, and each point outside it is removed. The algorithm also assumes the sensor system's orientation and altitude variables are provided.

LiDAR point clouds are obtained on the body's frame of reference. Due to the aerial vehicle's free-moving nature, the points need to be corrected for its angular movement (i.e. yaw, pitch and roll) to correctly access if the point is inside of the boundaries provided. This transformation consists of applying

a 3D rotation matrix to each point, and is represented as function $inertical\_transform$, in line 2 of Algorithm 1.

In the flight test scenario, the boundaries to eliminate non-desired detections are relatively straightforward and are composed of only two restrictions. The first restriction discards every point that is further away than a $70m$ radius from the sensor system. Since the target flew on an open field, most of the environment clutter was located outside this operational radius. The second restriction aims to remove ground returns. Knowing both the ground h_{ground} and sensor's altitude h_{sensor}, a world point $X = (x, y, z)$ is removed if

$$z \leq h_{sensor} - (h_{ground} + \delta h) \tag{3.1}$$

where δh is an altitude safety margin. As flight tests happened in a nearly flat farming field, δh was relatively small.

Algorithm 1 Pseudo-code for filtering the LiDAR's point cloud.

Input: LiDAR points, $point\_cloud$, sensors orientation and altitude, $inertial\_data$, $sensor\_altitude$ and environment boundaries $boundaries$.

Result: List with filtered LiDAR points, $filtered\_cloud$.

```
1 filtered_cloud= {};
2 corrected_cloud = inertical_transform(point_cloud, inertial_data);
3 for point in corrected_cloud do
4     if inside_boundaries(point, sensor_altitude, boundaries) then
5         insert point into filtered_cloud;
6     end
7 end
8 return filtered_cloud
```

3.3.2 Point Cloud Clustering

Since a single object can be responsible for multiple LiDAR returns, it is essential to create clusters around neighbouring points to discriminate between different targets. This task is performed by a density-based clustering solution called DBSCAN (Density Based Spatial Clustering of Applications with Noise) [61]. The algorithm requires two parameters: the maximum distance between two points for one to be considered in the neighbourhood of the other, and the minimum number of points to form a cluster. The first parameter is set to the largest dimension between flying targets, which from Table 4.4 is $0.5m$. The minimum number of points is set to one, which assumes the filtering step removes every point not of interest to the algorithm.

3.4 Calibration Procedure

This section goes over the methods used to find the intrinsic parameters of the camera (Section 3.4.1), as well as the steps taken to obtain the 3D-2D point correspondences to solve the PnP problem (Section 3.4.2).

3.4.1 Camera Intrinsic Calibration

The intrinsic camera parameters, mentioned in Section 2.4.1, need to be estimated as the solution for the PnP problem depends on it. A technique to estimate these parameters involves capturing sample images containing a well-defined pattern. A popular pattern used for intrinsic calibration is an 8×8 chequerboard pattern. The calibration procedure assumes each square's dimensions on the grid are given. It then obtains calibration points by extracting features related to square intersections. For accurate results, images must capture the calibration pattern at different distances and locations. The Matlab simple camera calibrator app [62] is used for this procedure, seen in Figure 3.4.

After the parameters are estimated, the algorithm projects the chequerboard points from world coordinates, obtained from the chequerboard pattern, into image coordinates. It then compares the reprojections to the corresponding detected points. The overall mean reprojection error obtained was 0.07 pixels, reflecting an accurate calibration.

As a side note, the chequerboard pattern was only captured at close range (i.e. $3m$ to $4m$) since it was too small to be captured at further distances. It is acknowledged that the calibration would present higher accuracy if the camera captured a larger pattern are longer distances.

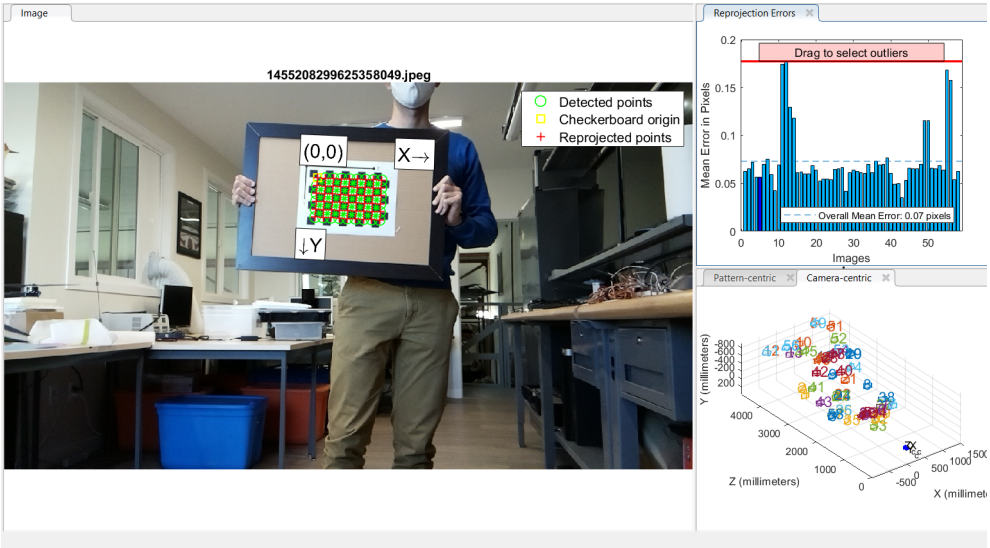


Figure 3.4: Sample of the Matlab simple camera calibrator app [62].

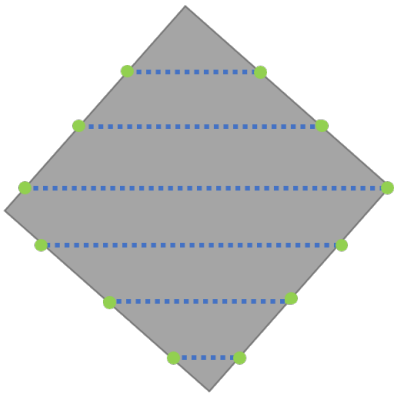
3.4.2 LiDAR-Camera Pose Estimation

The pose between the camera and the LiDAR must be estimated before LiDAR points can be projected into a camera frame. As mentioned in Section 2.4.2, this estimation problem is solved with the EPnP algorithm.

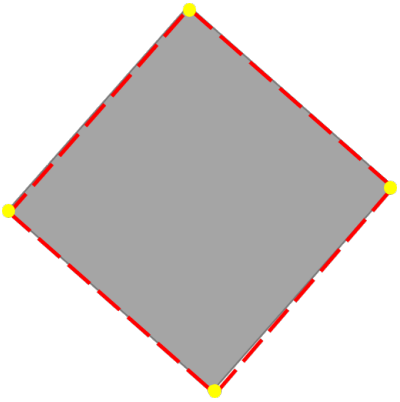
To find a solution, the EPnP algorithm requires the intrinsic camera matrix K, the radial and tangential distortion coefficients $(k_1, k_2, k_3, p_1, p_2)$, and several 3D-2D point correspondences.

Some literature approaches have used a calibration board's outer edges to obtain these correspondences [63]. The approach of this thesis follows a similar method by estimating the position of the calibration board's corners and using them as calibration points. If the rectangular board stays horizontal, the LiDAR scans are not able to accurately identify its height. However, by rotating the shape $45°$, as seen in Figure 3.5, LiDAR horizontal scans can intersect its four edges. Although simple in the camera frame, the LiDAR point cloud can not directly provide the location of its corners. So, their 3D coordinates have to be estimated.

First, 3D points have to be projected into a common plane, since LiDAR range measurements have deviations between them. A plane segmentation tool from Open3D [64] is applied to estimate the plane equation containing the calibration board. Every point is then projected into this common plane. Then, the algorithm achieves an estimate of the corner point's coordinates by computing the point cloud's convex hull and obtaining the minimum bounding box able to enclose it, as seen in Figure 3.5. However, this approach applied to a single board does not offer much depth distinction between corner points. So, to diversify calibration point locations, a total of four different board positions are captured, as seen in Figure 3.6. In total, the procedure extracted 16 pairs of 3D-2D point correspondences. The calibration board was captured at a maximum range of $9m$ since the point clouds were too sparse to detect the board in its entirety at further distances. More details regarding the LiDAR sensor used are given in Section 4.1.1.



(a) Repreentation of the LiDAR points, in blue, after the common plane projection, and its convex hull points in green.



(b) Repreentation of the minimum bounding box in red, and the estimated corner points in yellow.

Figure 3.5: Representation of the approach taken to estimate the 3D location of the calibration board corners.

Additional calibration points are obtained at further distances using a different approach. From the

flight test data, 3D-2D point correspondences are extracted by visual inspection, as presented in Figure 3.7. After a preliminary calibration procedure obtained from the previous method, LiDAR points were visually examined for where they were supposed to hit the target in the pixel context. In total, 22 point-pairs were extracted, ranging between $10m$ and $50m$.

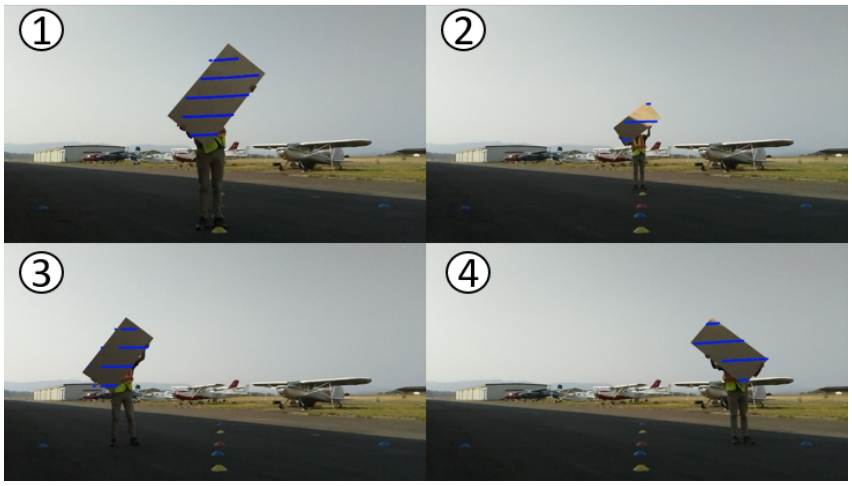


Figure 3.6: Reprojection of the point cloud into the calibration board at four different locations. The distance of each location is: 1 - $4m$; 2 - $9m$; 3 / 4 - $7m$.

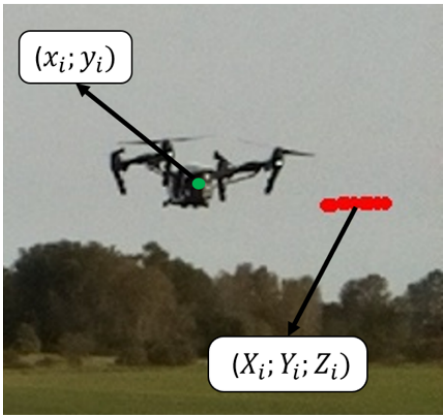


Figure 3.7: Example of the visual inspection method used to extract calibration points at further distances. A LiDAR point $P = (X_i; Y_i; Z_i)$ is visually associated with the pixel point $p = (x_i; y_i)$ for the i^{th} point correspondence.

3.5 ROI Creation

The goal of the region on interest (ROI) is to narrow the search area processed by the YOLOv3 detection algorithm. An overview of the ROI creation process is presented in Algorithm 2.

Each ROI has fixed dimensions to simplify its creation process. They are chosen based on the maximum pixel size that the target presents during the flight tests. The primary target measures $45cm \times 45cm$, as seen in Section 4.2.1. When captured by the camera at $10m$, it occupies an 80×50 pixel area

inside a 1280×720 image. Since the YOLOv3 detector requires an input size multiple of 32, the ROI size is set to 128×128 pixels. This restriction is related with its convolutional network architecture.

Approximate target locations come from two sources: LiDAR detections and Kalman filter predictions from the tracking algorithm. After the LiDAR point cloud is transformed into 2D clusters, a bounding box enclosing every point is created based on two values: the cluster centre and the maximum distance between two points. This bounding box is supposed to be representative of the captured target. However, LiDAR clusters may only capture a small portion of the target, leading to uncertainty regarding its outer limits. On the other hand, bounding boxes from Kalman filter predictions contain information on the target's pixel size, allowing a more informed decision on ROI placement. The function $cluster\_analysis$, line 8 of Algorithm 2, evaluates if the cluster centre is positioned inside one of the Kalman filter predictions. If so, it means the target is already being visually tracked, so the object is not added again to the objects of interest list. Although LiDAR clusters do not consistently provide target detections, they are a crucial mechanism for target acquisition.

After every object of interest is gathered, a ROI is created around each one. However, if an object is entirely enclosed by an existing ROI, this process is skipped. This simple solution presents similar problems to the regions proposals of the R-CNN algorithm (Section 2.1.1). This is because one object in the borderline of another ROI will create its own search region, leading to redundant computations. Additionally, overlapping ROI can cause the same target to be detected multiple times. As a solution, a non-maximum-suppression algorithm is applied to remove duplicate YOLOv3 detections. Given the present sUAS scenario with low vehicle density, the problem of overlapping ROI is not substantial.

Figure 3.8: Example of a frame processed by the sensor fusion algorithm. The LiDAR detections are the red dots, the ROI is the yellow bounding box, the ground truth is the light blue bounding box and the YOLOv3 detection is the dark blue bounding box.

In order to prepare the algorithm for the next frame, old objects of interest are discarded, and new ones are taken from the Kalman filter predictions. This step is represented between lines 20 and 24 of in Algorithm 2.

With the methods developed, the LiDAR acquires the presence of aerial targets, and the *YOLO-based tracker* detects and tracks their location in the image. Figure 3.8 represents a frame processed by the sensor fusion algorithm using the described approach. A LiDAR detection, the red dots, create the ROI, yellow bounding box, which is processed by the YOLOv3 detector, obtaining the target's visual detection, in dark blue.

Algorithm 2 Pseudo-code of the sensor fusion's main function .

Input: Sensor data from Camera, LiDAR , IMU and altimeter, $sensor\_data$, and environment boundaries for filter, $boundaries$.

Result: List of target trajectories, $trajectories$.

```
1  trajectories = {};
2  tracker = deepsort(); // Initialize tracker.
3  for image, point_cloud, attitude, altitude in sensor_data do
4      filtered_cloud = filtering(point_cloud, attitude, altitude, boundaries);
5      if length(filtered_clod) > 0 then
6          cluster_list = DBSCAN_cluster(filtered_cloud);
7          cluster_image_coordinates = 2d_projection(cluster_list);
8          objects_of_interest = cluster_analysis(cluster_image_coordinates, objects_of_interest);
9      end
10     regions_list = roi_creation(objects_of_interest);
11     yolo_detections = {};
12     for region in regions_list do
13         roi_image = crop_frame(image, region);
14         detections = run_yolo(roi_image);
15         insert detections into yolo_detections;
16     end
17     yolo_detections = non_max_suppression(yolo_detections);
18     update tracker with yolo_detections;
19     insert tracker.tracks into trajectories;
20     objects_of_interest = {};
21     predict tracker; // Predict tracks for future frame.
22     for track in tracker.tracks do
23         insert track into objects_of_interest;
24     end
25 end
26 return trajectories
```

Chapter 4

Experiments

This chapter presents the experiments performed. It starts by explaining the sensor system design and contruction (Section 4.1). It then describes the flight tests experiments (Section 4.2) and finishes by detailing how the flight operations were conducted and what data was collected (Sections 4.3).

This chapter will use the term remotely piloted aircraft (RPA) to refer to the aircraft flown during the flight tests. These RPA are considered to be sUAS. The RPA designation is mainly employed to make a distinction between autonomous aircraft.

4.1 System Assembly

This section details the payload design and construction. It gives an overview on the system components, architecture and final assembly.

4.1.1 M8 LiDAR Sensor

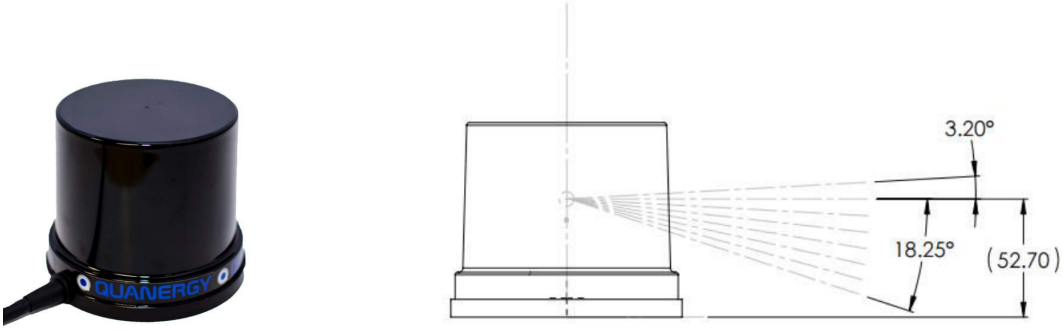
The LiDAR sensor used in the present work is the M8 LiDAR from Quanergy, shown in Figure 4.1a. The primary sensor attributes are presented in Table 4.1. The sensor weighs $940g$ and has a maximum range of over $100m$ at 80% reflectivity. While it's spinning, it can capture the environment at a rate between $5 - 20Hz$. The sensor presents eight vertical channels spread in a 21° FOV. The beams present an offset 3° upwards and 18° downwards, as presented in Figure 4.1b. The LiDAR sensor has a horizontal angular resolution between $0.03^\circ - 0.13^\circ$, depending on the framerate the sensor operates in.

Other works have used the M8 LiDAR showing it is well suited for 3D object detection despite having an inferior number of vertical channels compared to other LiDAR sensors, which can present 16 to 64 channels. Wei et al. [66] have employed the M8 LiDAR and a camera to detect beacons utilized to identify restricted areas.

In this work, the LiDAR sensor is turned upside-down to invert the direction of laser projection to $+18^\circ / -3^\circ$. This orientation is more suitable for aerial vehicle detection by avoiding to aim most of the laser beams downwards.

Table 4.1: Specification sheet of the M8 LiDAR sensor [65].

Parameter	Specification
Wavelength	$905nm$
Nominal Weight	$940g$
Maximum Range	$> 100m$ (80% reflectivity)
Range Accuracy $(1\sigma$ *at* $50m)$	$< 3cm$
Frame Rate	$5 - 20Hz$
Vertical FOV	$21° \, (+3° \, / \, - 18° \,)$
Horizontal FOV	$360°$
Vertical Channels	8
Angular Resolution	$0.03° \, - 0.13°$ (dependent on frame rate)
Output Rate	420,000 points per second



(a) Image of the M8 LiDAR from Quanergy.

(b) M8 LiDAR vertical beam angles.

Figure 4.1: Visualization of the M8 LiDAR appearance and vertical laser beam angles [65].

4.1.2 Onboard Computer Optimization

It is a challenge to guarantee a constant framerate for all sensors while recording, mainly caused by the onboard computer's limitation. The main constrain is the high-volume sensor data captured, which quickly overwhelms CPU resources. Additionally, writing data in memory is limited as the computer cannot manage large data bandwidths.

Firstly, to increase available computational resources, the latest Raspberry Pi 4 was chosen as the onboard computer. Also, while experimenting with several camera alternatives, the Pi camera module was seen as the option to optimize data capture better. In contrast with other USB webcams, it uses the graphics processing capabilities of the Broadcom CPU. The Pi camera can process image data at higher framerates, saving CPU resources for other concurrent tasks.

Furthermore, to lower data bandwidth and spare storage capacity, camera frames are saved as a compressed *JPEG* file format instead of the raw sensor data, reducing the bandwidth from $83.13Mb/s$ to $1.80Mb/s$ for images with 1280×720 resolution. Similarly, knowing the sensor fusion methods would only need LiDAR points inside the FOV of the camera, settings were adjusted on the M8 LiDAR as to only output points inside an $80°$ FOV, instead of the standard $360°$. This procedure decreased LiDAR data bandwidth from $13.90Mb/s$ to $3.05Mb/s$.

4.1.3 ROS Environment

Synchronization between every sensor is essential to correlate observations reliably, even if they operate at different framerates. The onboard computer uses the Robotic Operative System (ROS) [67] to provide an environment for hardware abstraction capable of sensor unification.

Every process inside ROS is called a node, and nodes can communicate with each other through topics. Topics are named buses, and messages are exchanged by subscribing or publishing to a specific topic. Each topic message has an associated ROS timestamp, necessary for synchronizing sensor data on post-processing.

Additionally, ROS provides a tool called ROSBAG, which can reliably record topic messages to be post-processed on the ground. It preserves vital information by avoiding the deserialization and reserialization of messages. Thus, while playing back recorded data, message timestamps are maintained.

A diagram of how the developed ROS architecture handles sensor data is represented in Figure 4.2. The data from each sensor is received by a ROS node, which processes and publishes it into a ROS topic. Those topics are then recorded into a ROSBAG.

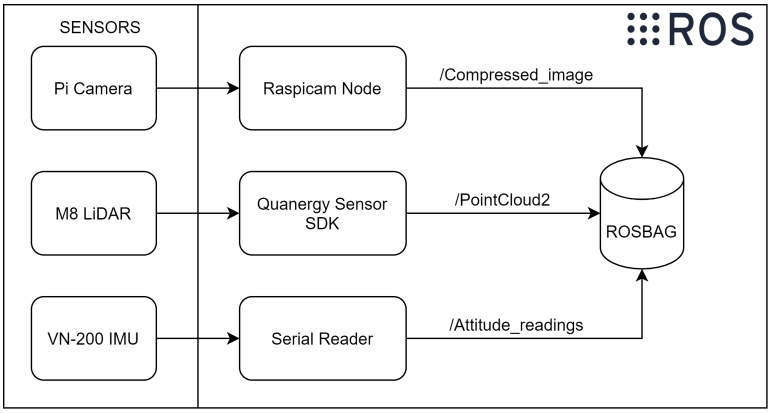


Figure 4.2: Diagram of the developed ROS architecture. The sensors are represented on the left, while the ROS environment is represented on the right, which shows the various ROS nodes that process sensor data and the ROSBAG that records the ROS topics.

In an ideal scenario, every sensor would be recorded in an identical framerate $(20Hz)$, with corresponding sensor frames presenting the same timestamp. However, this is not a trivial task to accomplish. The presence of jitter on sensor data streams allied with difficult timing cues to begin the recording processes between every sensor makes it almost impossible for them to be in sync at the same framerate. The approach taken opted to maximize the camera and IMU sensors' framerate while maintaining the LiDAR at $20Hz$. Then, on post-processing, extra frames would be trimmed to present a consistent output of $20Hz$ on all sensors. The trimming process chooses sensor frames with the smallest timestamp deviations from LiDAR messages.

This approach is not ideal, as a target might move between corresponding frames. For example, the camera is recorded at $30Hz$, presenting a $33ms$ gap between frames. The worst-case scenario is when a LiDAR frame is precisely between two camera frames, meaning the largest time deviation between a LiDAR and camera frame is $16.7ms$. Similarly, the IMU is recorded at $40Hz$, making this maximum

difference between it and LiDAR frames of $12.5ms$.

4.1.4 Communications

The ground station needs to have a communication link with the sensor system while flying. Remote tasks include memory management and operating the sensors and the actuators. A connection through a Secure Shell (SSH) protocol provides access to the onboard computer's Linux environment. However, while it might easily work in the lab, there is no available network to allow this connection on the flight test field. Thus, a digital data link (DDL) over radio between the ground station and the sensor system is created, guaranteed by a wireless connection between two pMDDL900 radio devices in a master/slave configuration. They are able to provide the IP abstraction needed for a SSH connection. Figure 4.3 represents the pMDDL900 radio device flown on the payload.

When connecting both radio devices, care must be taken to ensure the path loss (the reduction of signal strength from the transmitter to the receiver) between equipments does not exceed the communication system's gain. The module belonging to the ground station was positioned in the test van's top section to maximize the signal strength.

4.1.5 Power System

The sensor system is mounted onboard a DJI Matrice 600. While airborne, the sensor system's power supply must be independent of the aircraft, as not to compromise its safety.

Figure 4.3 presents a layout of the sensor system components. Each payload component has its own constraints on what input voltage it requires, as presented in Table 4.2. Devices like the servo, Pi Camera and IMU can be powered directly through the Raspberry Pi, only requiring it to be powered with a 5V input voltage. The pMDDL900 radio can work at a wide range of input voltage. On the sensor system, the biggest constrain is the M8 LiDAR power supply, which requires an input voltage of 24V with a tolerance of $\pm 1V$.

Table 4.2: Payload components input voltage and power consumption while recording sensor data.

Device	Input Voltage [V]	Power Consumption [W]
M8 LiDAR	24 ± 1	17
Raspberry Pi + Peripherals	5 ± 0.25	5.2
pMDDL900	19 ± 11	3

The energy solution is composed of two LiPo 6S 1200 mAh batteries connected in parallel to double battery capacity to 2400 mAh, while maintaining the same voltage. Being a 6S battery, it has an output voltage of 25.2V when fully charged, dropping to 23.1V at 50% capacity. Thus, the battery pack can directly power the LiDAR and the DDL radio. Additionally, a DC-DC converter is used to power the Raspberry Pi, lowering the supplied voltage to 5V. It is estimated that the battery pack can support the sensor system for up to 38 minutes while recording, which is more than the DJI Matrice's battery life, which will carry the payload. In experiments where the payload is not flying, the system is externally powered by a 10,000 mAh 6S battery to spare the onboard battery life.

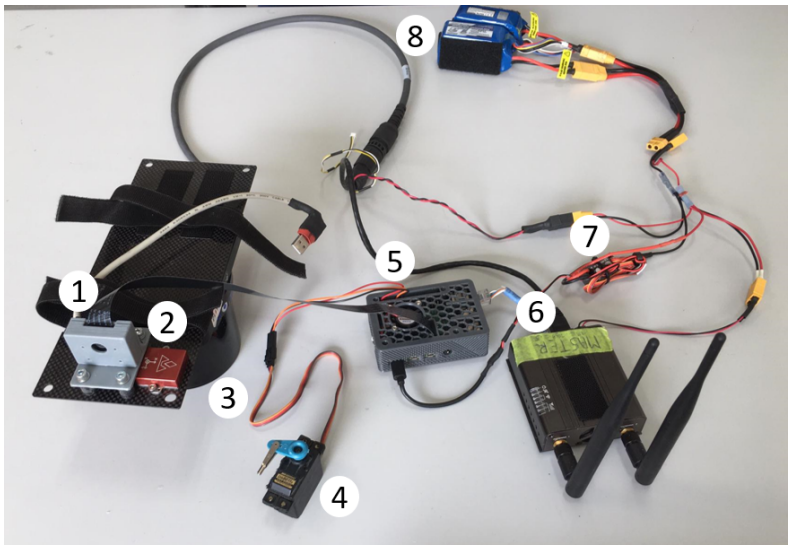


Figure 4.3: Sensor system components layout: 1. Pi Camera; 2-VN-200 IMU; 3 - M8 LiDAR; 4 - Servo; 5 - Raspberry Pi 4; 6 - pMDDL900 radio; 7 - DC-DC converter; 8 - Battery pack.

4.1.6 Mechanical Assembly

The mechanical assembly aims to attach the sensor system safely to the DJI Matrice 600. The payload should not compromise the aircraft's safety or stability. The payload CAD model and the fully assembled structure onboard the DJI Matrice 600 is shown in Figure 4.4.

(a) CAD model of the payload.

(b) Payload assembled onto the DJI Matrice 600.

Figure 4.4: Sensor system payload design and assembly.

The sensor system attaches to a carbon fibre base plate and is protected by a 3D printed case, which encloses the system. The case presents two openings on the top section for the radio antennas. Additionally, it has one in the front for the camera and another in the back for the power cables.

The payload attaches to the RPA through two arm structures, also 3D printed. These structures connect to the LiDAR heat shield plate via two steel shoulder screws, which create an axis of rotation.

The servo is mounted into the right arm structure and controls the pitch motion via a mechanical link to the case to perform the pitch angle experiments explained in Section 4.2.3. The payload's centre of gravity (CG) needs to be under the rotation axis to minimize torque on the servo created by the payload weight.

Overall, the payload weights 2.97 kg. Its distribution across the components is presented in Table 4.3. A close inspection unveils that 45% of the total weight is attributed to the LiDAR sensor and its heat shield. The heat shield provides important heat absorption to the LiDAR sensor. Experiments showed that the LiDAR would quickly overheat without this metal plate. Despite the added weight, the payload is under the 5.5 kg payload limit of the DJI Matrice 600.

Table 4.3: Weight distribution of the payload components.

Component	Weight [g]
M8 LiDAR	932
Structure and Cables	675
LiDAR Heat Shield	400
2 x 1200mAh Batteries	400
pMDDL900	240
VN-200 IMU	164
Raspberry Pi 4	94
Servo	63
Total	2968

Finally, to stabilize the Pi Camera and isolate it from vibrations induced by the flying vehicle, the camera case is mounted on top of vibration-damping sandwich mounts, made from rubber, seen in Figure 4.3.

4.2 Flight Test Experiments

This section presents the remotely piloted aircraft (RPA) deployed for the flight test operations (Section 4.2.1) and explains the three main experiments performed.

A simplified diagram of the experiments is presented in Figure 4.5. On it, the three main experiments are highlighted: *moving target*, *moving sensor* and *multi-target free-flight*. Each of the first two is divided into two different experiments related to the sensor system position and movement. For the *moving target* experiments, the sensor is either position on the ground or in a hover position. In the *moving sensor* experiments, the sensor performs two different manoeuvres: altitude and pitch angle variation.

4.2.1 RPA Deployed

Three different RPA flew on the flight tests. The main attributes of each aircraft is presented in Table 4.4. The aircraft carrying the sensor payload was the DJI Matrice 600 Pro, presented in Figure 4.6. Primarily designed to fly heavy equipment for professional-level cinematography, it is the largest aircraft of the three and provides an ideal platform to support the sensor system. With a maximum take-off weight of 15.5 kg, the RPA has an endurance of 18 minutes while carrying a 5.5 kg payload.

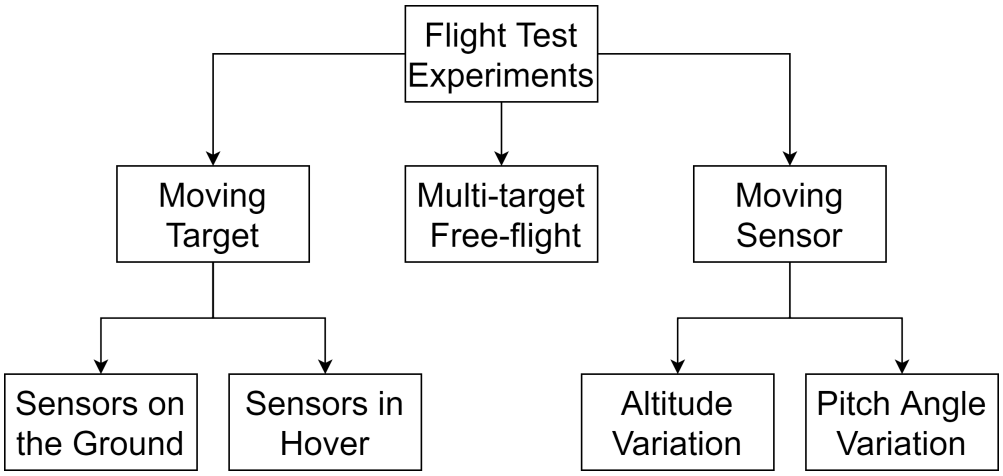


Figure 4.5: Diagram of the flight test experiments.

Figure 4.6: DJI Matrice 600 carrying the sensor system.

Figure 4.7: DJI Inspire being flown as the primary target.

Figure 4.8: DJI Mavic being flown as the secondary target.

The two remaining RPA were operated as flying targets, presented in Figure 4.7 and 4.8. The DJI Inspire and DJI Mavic weight 2.8 kg and 0.7 kg, respectively, corresponding to the micro and mini UAS categories explained in Section 1.2. They both are popular commercially available sUAS, becoming ideal test vehicles for C-sUAS experiments. Between them, the DJI Inspire has the largest size, and for that reason, it was considered the primary target, flown on all experiments. The DJI Mavic was only used during the *free-flight* experiments, where all RPA were flown simultaneously.

Table 4.4: Properties of each flying vehicle used during the flight tests.

	DJI Matrice 600 Pro	**DJI Inspire**	**DJI Mavic**
Dimensions [cm]	167x152x73	44x45x30	32x24x8 (unfolded)
Weight [g]	10,000	2,845	734
Nº of Motors	6	4	4
Autonomy [min]	18 (w/ 5.5 kg payload)	18	21
Battery	LiPo, 2S, 6000 mAh	LiPO, 6S, 4500 mAh	LiPO, 3S, 3830 mAh

4.2.2 Moving Target Experiments

The *moving target* experiments were performed while the sensor system is stationary relative to the surrounding environment. Two different sensor locations were studied: on the ground level and in hover. While in these positions, it would capture the RPA target performing a manoeuvre named *cross manoeuvre* at different distances. Figure 4.9 presents a visual representation of this manoeuvre.

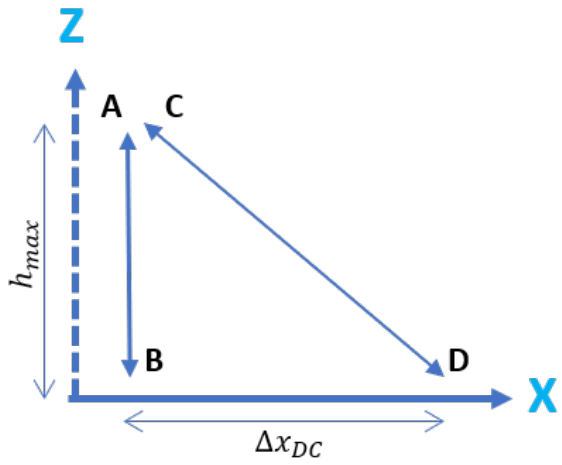


Figure 4.9: *Cross manoeuvre* diagram, with same frame of reference as in Figure 4.10. The target begins by moving between A-B-A, followed by C-D-C.

A concerning limitation of the M8 LiDAR, as presented in Section 4.1.1, is its sparse vertical resolution. The manoeuvre was created to increase the probability of the target being detected by the LiDAR. To do so, the RPA performs significant altitude variations to cross as many horizontal scanning lines as possible. For this reason, it is named the *cross manoeuvre*. In its essence, the manoeuvre consists of simple climb and descents, performed both vertically and diagonally. The objective of the diagonal movement is to diversify the horizontal detection location. The *cross manoeuvre* can be split into four different steps, presented in Figure 4.9. The target starts at point A and begins by performing a vertical descent ($A - B$) and climb ($B - A$). It then transitions into the diagonal section, once again descending ($C - D$) and climbing ($D - C$). The RPA must move along the $X - axis$ to keep itself at a constant horizontal distance from the sensors, as seen in Figure 4.10. The *cross manoeuvre* is then performed at different range intervals to evaluate the relationship between LiDAR detections and target distance.

The *cross manoeuvre* has two parameters: the maximum height h_{max} and the horizontal distance Δx_{DC} between point D and C. To allow the drone a chance to pass over every possible horizontal scanning line, the maximum height h_{max} was different at each distance d, taken from the equation

$$h_{max} = h_s + d \cdot tan(\alpha_{max}) \tag{4.1}$$

where h_s is the height of the sensor system and α_{max} is the highest elevation angle the LiDAR is able to scan.

The RPA target performs the *cross manoeuvres* between $10m$ and $60m$ from the sensor system.

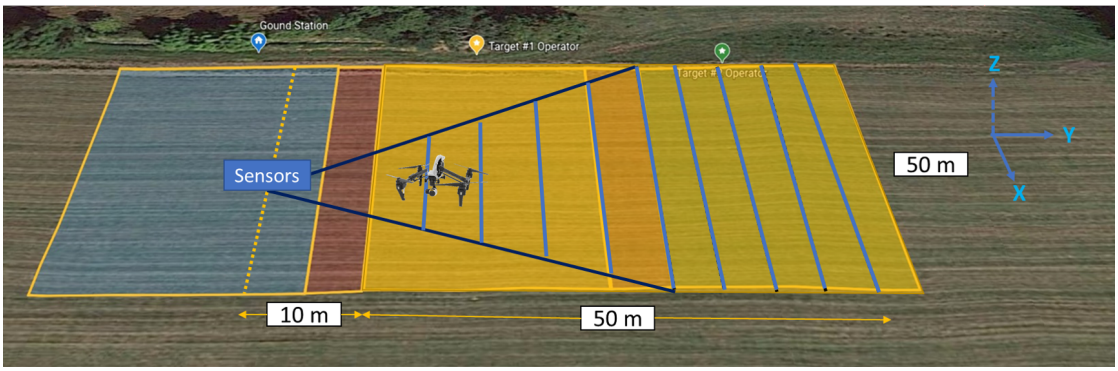


Figure 4.10: Visual representation of the *moving target* experiments. The DJI Inspire performs the *cross manoeuvre* along the light blue lines, staying parallel to the $X - axis$. The dark blue lines represent the FOV limits of the camera. The sensor system has the same X and Y values in every manoeuvre.

Ideally, Δx_{DC} should be set as always to keep the RPA within the camera FOV. As seen in Figure 4.10, markers were physically placed on the field to help the pilot-in-command visualize these boundaries.

During the experiments, the sensor system was kept at the same X and Y positions. Its altitude was $h_s = 1m$ while on the ground, and at $h_s = 10m$ while in hover.

4.2.3 Moving Sensor Experiments

The added mobility generated by attaching the sensor system onboard an aircraft provides new sensing opportunities, possibly detecting stationary targets that otherwise would not be visible in the LiDAR sensor, i.e. between horizontal scans. In the *moving sensor* experiments, the RPA target is positioned in such regions in the hover state. The benefits of sensor mobility are evaluated by exploring two different techniques: altitude and pitch angle variation.

The first manoeuvre explores the sensor's altitude variation. As the LiDAR altitude changes, the horizontal scanned regions can overlap each other by moving high enough, guaranteeing the sensor performs a complete search of the airspace. Thus, the height Δh between two consecutive scans at a certain range represents the same vertical displacement the LiDAR needs to achieve and is given by:

$$\Delta h = d \cdot (tan(\alpha_{i+1}) - tan(\alpha_i)) \tag{4.2}$$

where d is the horizontal distance between the target and the sensor, α_i is the angle between the horizon and the i^{th} horizontal scan. One obvious concern with this approach is that as α_i and α_{i+1} increase, so does the height Δh. For the M8 LiDAR, the separation between horizontal scans is $\Delta\alpha = \alpha_{i+1} - \alpha_i = 3°$, and the elevation of the two highest horizontal arrays is $\alpha_8 = 18.25°$ and $\alpha_7 = 15.25°$. So, for a $d = 60m$, an altitude variation of $\Delta h = 3.42m$ would be necessary to guarantee an overlap between scanned regions. Experimentally, the sensor system performs $\Delta h = 5m$, theoretically achieving no blind regions up to $d = 87.8m$.

The second technique examines how the pitch angle movement of the LiDAR can be used to detect other RPA. Tilting the sensor system pitch angle θ more than the angle $\Delta\alpha$ between the LiDAR horizontal

scans allows the horizontal scans to overlap, creating a thorough inspection of the airspace for targets. Experimentally, precise pitch angle motion of the sensor system is performed by a servo-controlled mechanism, as discussed in Section 4.1.6. This solution preserves the aircraft's position and stability. The experiments perform a pitch variation of $\Delta\theta = 5°$, guaranteeing an overlap between LiDAR scans, since $\Delta\alpha = 3°$.

The target is positioned at the centre line position in both experiments, along the Y-axis, as demonstrated in Figure 4.11. Like the *moving target* experiments, the vehicle moves between $10m$ and $60m$ from the sensor system. At each segment, the target sustains a hover manoeuvre at different altitudes to keep it at a constant $10°$ elevation angle relative to the payload. The sensor system preserves the same X and Y values for the entire *moving sensor* experiments, with the altitude varying between $10m \leq h_s \leq 15m$ on the altitude variation experiment. On the pitch angle experiment, the system hovers at $h_s = 10m$.

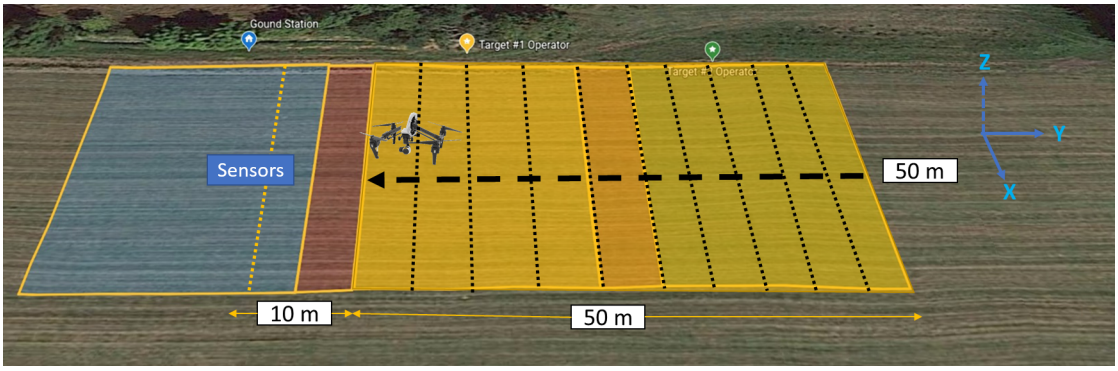


Figure 4.11: Visual representation of the *moving sensor* experiments. The sensor system has the same X and Y values while performing the altitude and pitch angle experiments. The DJI Inspire stays on top of the $Y - axis$, at different distances from the sensor system, in a hover position.

4.2.4 Multi-target Free-Flight

The objective of the free-flight experiments is to capture realistic flight test data. Each agent should have no predefined trajectories or manoeuvres. Nonetheless, the hazard of operating multiple RPA within share airspace raises safety concerns. As such, each RPA has a specific airspace region assigned to itself, with 'no-fly' zones in-between, as represented in Figure 4.12. The DJI Mavic, being the aircraft with the smallest dimensions, is placed in the airspace closest to the sensor system. Additionally, RPA pilots are placed inside the 'no-fly' regions to prevent the RPA from crossing to unwanted regions.

4.3 Flight Test Operations

The flight test operations were performed by a team of specialists from the Centre for Aerospace Research (CfAR) of the University of Victoria, Canada. A total of eight crew members contributed to the flight test experiments, from which six had a drone pilot licence and an extensive experience in flight operations.

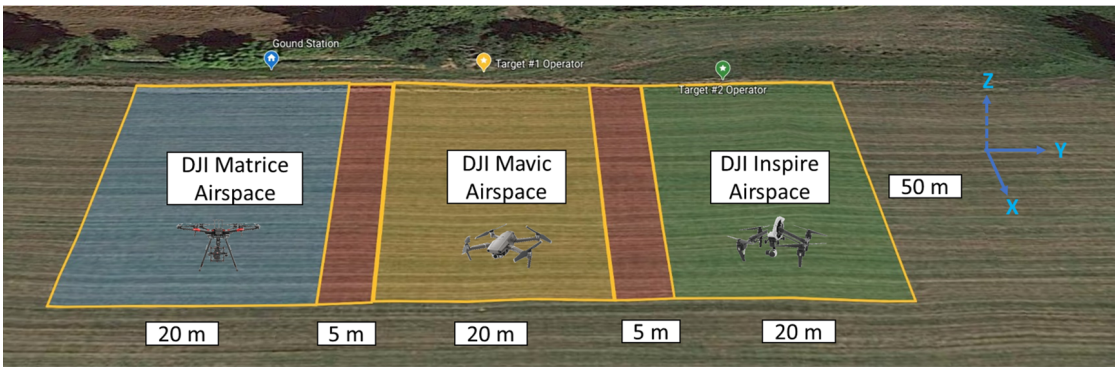


Figure 4.12: Unique airspace regions assigned to each RPA for the *free-flight* experiments. The DJI Matrice 600 carries the sensor system payload, while the DJI Mavic and DJI Inspire act as flying targets.

4.3.1 Regulations and Test Crew Roles

The testing field for this research was conducted within restricted airspace, which required the CfAR test pilots to follow the procedures for an RPA Advanced Operation. This included prior registration of each aircraft, Advanced Operators certificates issued for each pilot, authorization for the operation from NAV Canada, and permission from the local air traffic management.

During the flight tests, the following roles were assigned to flight test crew members:

- **Pilot-in-command**: main body responsible for the control and safety of the RPA. In total, three licenced pilots were responsible for the RPA operations (Section 4.2.1).
- **Visual Observer**: assists the pilot in ensuring the safe conduct of a flight under visual line-of-sight.
- **Range and Safety Officer**: monitors any external threats that may come into the airspace, which might include incoming aircraft or pedestrians.
- **Flight Director**: responsible for the coordination of flight test procedures between crew members.
- **Ground Control**: manages and controls the sensor system payload.

4.3.2 Data Collected

The flight manoeuvres performed were not as extensive as initially planned due to operational time constrains and shorter than expected drone battery life. Nonetheless, valuable data was captured, resulting in a successful flight test, performed between 2 pm and 6 pm on September 30, 2020. A photo taken during the flight test operations is presented in Figure 4.13.

Every target manoeuvre was performed between the $10m$ and $60m$ range in increments of $10m$. The *free-flight* experiments were recorded for 4 minutes and 20 seconds, resulting in $5,179$ captured frames. In total, $23,425$ frames, distributed across 29 manoeuvres, were captured and manually labelled for their ground truth. This process was done on a Matlab video labeller application [62].

Figure 4.13: Photo taken during the flight test operations. It presents the test van on the left, followed by the flight test crew on the bottom centre. Finally, two RPA are represented on the top right, the DJI Matrice 600 and DJI Mavic.

Chapter 5

Results

This chapter presents the results obtained during the flight test experiments explained in Section 4.2. It starts by presenting the results relative to the LiDAR detection performance (Section 5.1), followed by the accuracy of the extrinsic calibration (Section 5.2). The chapter then approaches the results from the visual detection methods (Section 5.3), as well as a comparison with the sensor fusion methods (Section 5.4). The chapter finishes by discussing the results (Section 5.5).

All the metrics presented in this chapter are explained in detailed in Section 2.3.

5.1 LiDAR Detection Performance

This section presents the LiDAR detection results on a sUAS, specifically, the DJI Inspire. It is divided into two subsection: *moving target* and *moving sensor* experiments.

Additionally, all the data presented reflects the point cloud after the filtering step, seen in Section 3.3.1. The filtering method showed to be extremely effective for the flight test scenario presented. So, the following results reflect a point cloud without false positives.

5.1.1 Moving Target Experiments

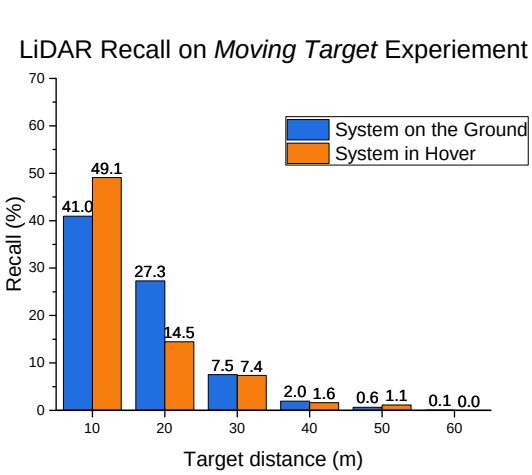
As explained in Section 4.2.2, the *moving target* experiments had the sensor system stationary relative to the surrounding environment, either on the ground or in a hover position. The target performed the *cross manoeuvre* at different distances.

Figure 5.1a shows comparable recall results between the LiDAR on the ground or in a hover position. Both scenarios show a drastic decrease in target recall throughout the $60m$ range. The LiDAR presents almost no detections at this last distance, except for a single point captured in the ground position. Although comparable, there are discrepancies between both scenarios, more predominant at closer ranges.

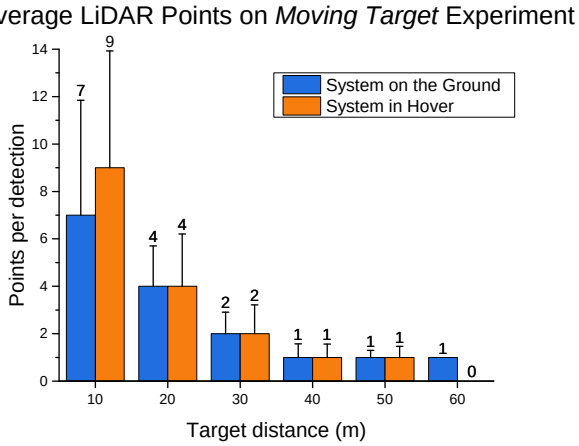
Figure 5.2b represents multiple LiDAR returns being generated for a single target for distances less than $30m$. However, for distances beyond $40m$, the RPA is mainly captured by a single LiDAR point. A

cluster formed by a single point can be problematic to confidently evaluate as a real target. Nevertheless, it might be plausible if there is enough confidence in the filtering step to remove false positives.

These results display the LiDAR sensor's inability to continuously capture the target location and act as a single sensing solution on a static platform. The discrepancies observed between the ground and hover experiments are not very significant. They seem to indicate no meaningful performance deterioration induced by the LiDAR sensor's presence onboard an aircraft, compared to having it on the ground position.



(a) LiDAR recall with target distance.

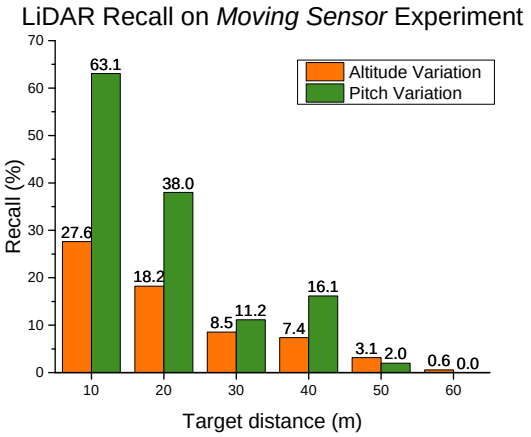


(b) LiDAR points per detection with target distance.

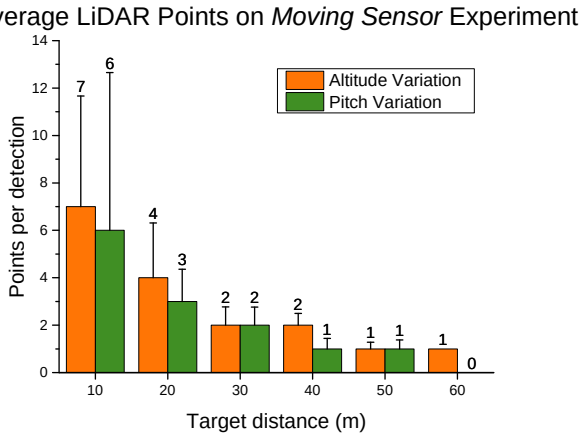
Figure 5.1: LiDAR recall and number of points per detection on the *moving target* experiments.

5.1.2 Moving Sensor Experiments

The LiDAR detection performance on the *moving sensor* experiments (Section 4.2.3) is analyzed similarly to the methods applied in the previous Section 5.1.1. Once again, the LiDAR's recall and average number of points per detection is evaluated, as shown in Figure 5.2.



(a) LiDAR recall on *altitude variation* and *pitch variation* experiments with target distance.



(b) Points per detection on *altitude variation* and *pitch variation* experiments with target distance.

Figure 5.2: LiDAR recall and number of points per detection on the *moving sensor* experiments.

The *moving sensor* experiments proved to be more consistent than the *moving target*. For example, the sensor's pitch movement is controlled by the onboard computer, granting a consistent and precise 5° pitch variation. Additionally, the sensor system's altitude varied only between $10m$ and $15m$ while maintaining the same X and Y positions. These techniques produced fewer uncertainties than the more complex *cross manoeuvre* of the previous experiments.

Analysing the altitude variation experiment in Figure 5.2a, it is observed a 27.6% target recall at $10m$, which is inferior to both *moving target* experiments. However, recall consistency is improved, as it only drops below 3% over the $50m$ range, while previous results show a drop below 3% at $40m$. Additionally, target recall values on the *pitch variation* experiment show the highest recall rates of all approaches, achieving 38% recall at distances up to $20m$, only dropping below 10% at ranges greater than $40m$.

Observing Figure 5.2b, both manoeuvres present similar trends in the number of points per detection and are on par with the *moving target* experiments. Multiple points per detection are presented until the $30m$ range, dropping only to a single point per detection above this threshold. The exception is the *altitude variation* experiment, which presents two points instead of one at $40m$.

These results reinforce the evidence of the LiDAR's drastic performance degradation with the increase in target distance. They also demonstrate the LiDAR's potential to detect hovering sUAS through both dynamic movements.

5.2 Extrinsic Calibration Accuracy

This section evaluates the pose estimation accuracy. The calibration accuracy is related to its ability to project 3D LiDAR target points inside the 2D ground-truth label. The projected point can either land 'inside' or 'outside' the ground truth bounding box. Additionally, a 'near' label is adopted to specify if a point lands in the proximity of the ground truth. This classification is reserved for points that land in an area with twice the ground truth's height and length.

As explained in Section 3.4.2, two different techniques to capture 3D-2D point correspondences were explored. The first was by extracting the corner points of a calibration board. The second extracted point correspondences by visually estimating where each LiDAR point would correspond on the target's pixel location.

Table 5.1 presents a comparison between PnP solutions obtained with both solutions on the *moving target* experiments, with the additional scenario of combining every captured point to solve the PnP problem. The *moving sensor* experiments are not analyzed in this section, as the target is positioned consistently at the centre line, as explained in Section 4.2.3. Thus, using the data from this experiment would not be representative of the calibration accuracy across the entire image.

Table 5.1: Extrinsic calibration accuracy on the *moving target* experiments. Three point extraction methods are used to solve the PnP problem: calibration board, visual inspection and both combined. Each solution presents three labels for a LiDAR point projection: 'inside', 'near' or 'outside' the target's ground truth. These results are shown for the two *moving target* experiments: on the ground and in hover.

		Calibration Board			Visual Inspection			Both Combined			
	Target Distance [m]	Inside	Near	Outside	Inside	Near	Outside	Inside	Near	Outside	Total Points
System in Hover.	**10**	514	725	124	981	306	76	**1127**	**160**	**76**	1363
	20	0	275	480	695	39	21	**708**	**26**	**21**	755
	30	0	98	197	295	0	0	295	0	0	295
	40	0	4	39	41	2	0	41	2	0	43
	50	0	0	22	**22**	**0**	**0**	17	5	0	22
System on the Ground.	**10**	1254	573	0	1674	153	0	**1772**	**55**	**0**	1827
	20	2	857	339	1169	29	0	**1171**	**27**	**0**	1198
	30	0	10	194	**204**	**0**	**0**	203	1	0	204
	40	0	0	35	35	0	0	35	0	0	35
	50	0	0	13	13	0	0	13	0	0	13
	60	0	0	1	0	0	1	0	0	1	1

As a side note, more points at close ranges are not synonymous with a higher number of ground truth boxes. It is just the effect of a higher number of points per detection, discussed in the LiDAR detection analysis of Section 5.1.

The PnP solution obtained with the *calibration board* shows admissible accuracy results at close ranges. They also present an inability to transform any target points over $40m$ correctly. This accuracy drop with distance was expected, considering the distance between the sensor and point correspondences varied between four and nine meters. Thus, slight errors between 3D-2D point correspondences at these distances would become meaningful deviations at longer ranges.

The PnP solution obtained with the *visual inspection* method shows higher accuracy results across all ranges. Still, multiple projected points between $10m$ and $20m$ are classified as 'near' the ground truth. Accuracy at close ranges improved by combining all point correspondences to solve the PnP problem. Multiple points previously labelled as 'near' the ground truth improve to become labelled as 'inside'. However, it came at the cost of a slight loss in accuracy for the hover experiment at $50m$. Also, no PnP solution could show accurate results at the $60m$ range.

It is observed that the number of points 'outside' the ground truth does not vary between the *visual inspection* and *both methods combined*. These are points transformed out of the image frame bounds when the target is manoeuvring near the image's edge. Additionally, they appear to be much more significant in the hover rather than in the ground experiment.

Finally, the calibration accuracy for the *free-flight* experiment is presented in Table 5.2. Similar conclusions are taken regarding the three PnP solutions. The best accuracy is presented when a combination of all point correspondences is used.

Table 5.2: Extrinsic calibration accuracy on the *free-flight* experiment. Three point extraction methods are used to solve the PnP problem: calibration board, visual inspection and both combined. Each solution presents three labels for point projections: inside, near or outside the target's ground truth.

Point Extraction Method	Inside	Near	Outside	Total
Calibration Board	0	115	1243	1358
Visual Inspection	1322	36	0	
Both Combined	1344	14	0	

5.3 Visual Methods Results

This section presents the visual detection and tracking results obtained on the *moving target* and *free-flight* experiments. It starts by presenting the performance of the YOLOv3 detector (Section 5.3.1), followed by the results of the *YOLO-based tracker* (Section 5.3.2), where its ability to follow sUAS is evaluated. The results from the *moving sensor* experiments are excluded from this section as the target is in hover on the central part of the image. Therefore, it is not considered to be an interesting target to track.

5.3.1 YOLOv3 Detector

The YOLOv3, discussed in Section 3.2.1, is the algorithm responsible for detecting sUAS targets. A precision-recall (P-R) curve evaluates its ability to detect and locate targets captured during flight testing. Section 2.3 describes the concept of a P-R curve in further detail. The non-maximum-suppression parameters of the YOLOv3 algorithm were set with an IOU threshold of 50% and a minimum confidence threshold of 0.01. Lower confidence values proved to be problematic for the algorithm, hence its value. The P-R curve obtained is presented in Figure 5.3, showing an average precision (AP) of 32.0%. The AP serves as an overall metric to describe the detector's performance. However, the value obtained is low, and a close inspection of the P-R curve indicates the probable causes.

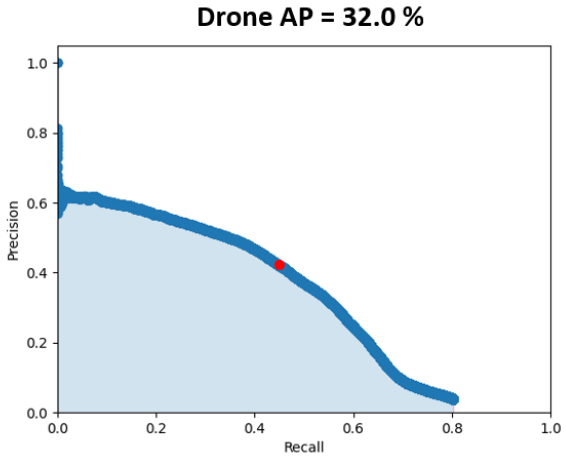


Figure 5.3: Precision-recall curve for the YOLOv3 detector. The presented results are based on the entire flight test data collected. The red point represents the highest $F1_{score}$ achieved of 0.44.

An ideal P-R curve would present a constant precision value of 100% as the target recall increases, presenting a horizontal line for all confidence threshold values. The negative slope in Figure 5.3 indicates an increasing proportion of false positives with a decrease of the confidence threshold. On examining detection samples presented in Figure 5.4, the various types of false targets collected become apparent. Obvious wrong detections, like the treetop, expose flaws in the training process. Other false positives present background clutter that is difficult to distinguish between a true and a false positive, even for the human eye. This is because the target and clutter can appear very similar. The bottom right sample in Figure 5.4 represents a safety cone, which is fairly similar to the true positive above. However, when the target is above the horizon, its features contrast more with the simple background, leading to more consistent detections.

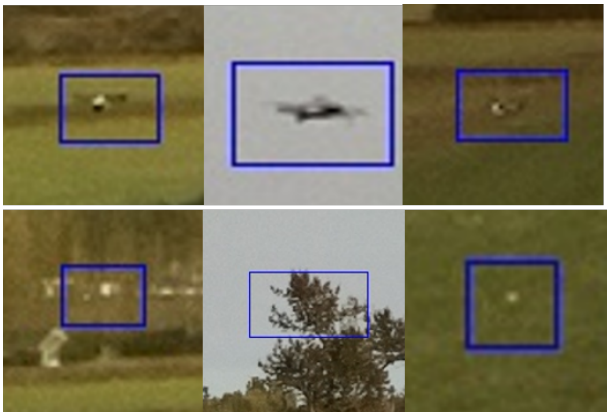


Figure 5.4: YOLOv3 detection examples. The top row presents true positives, and the bottom row false positives.

Additionally, with a minimum confidence threshold of 0.01, the detection algorithm could only achieve a recall of around 80%, contributing to the low AP number. A closer inspection reveals that the vast majority of the non-detected targets happen on occasions where the RPA is far and in locations with complex backgrounds, seen in Figure 5.4.

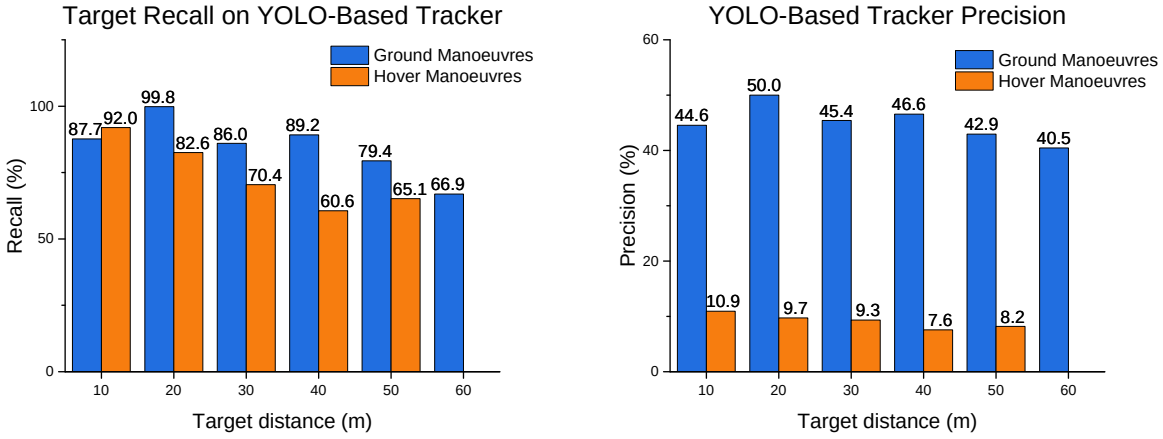
Furthermore, the camera frames are captured in the compressed *JPEG* format, as explained in Section 4.1.2. This diminishes image quality and makes it more challenging for the detector to extract small target features. In addition, the dataset used to train YOLOv3 was captured by a ground-based system. Thus, the vehicles are predominantly observed above the horizon. This would not prepare the network for scenarios with sUAS against textured backgrounds.

Finally, to determine with which confidence threshold the algorithm would perform best, the P-R curve's point is chosen based on the $F1_{score}$. This metric, described in Section 2.3, provides a balanced representation of the algorithm's precision and recall. The best $F1_{score}$ is 44%, obtained with a confidence threshold of 0.26. This point, represented by a red dot in Figure 5.3, has a precision of 42% and a recall of 46%. The following results present the YOLOv3 detector set on this confidence threshold.

5.3.2 YOLO-Based Tracker

This section evaluates *YOLO-based tracker's* performance based on the *moving target* and *free-flight* experiments. This algorithm is explained in Section 3.2.

The *moving target* experiments provide a valuable platform to analyze the tracking capabilities across multiple target distances. Additionally, the comparison between the ground and hover manoeuvres allows evaluating the effects of mounting the sensor system on a flying platform. Figure 5.5 presents obvious contrasts between ground and hover experiments. The *YOLO-Based tracker* presents a superior performance when the system is on the ground, having average higher recall and precision values across multiple target distances.



(a) Recall of *YOLO-based tracker*.

(b) Precision of *YOLO-based tracker*.

Figure 5.5: *YOLO-Based tracker* on the *moving target* experiments.

Both manoeuvres in Figure 5.5a present a recall decline with an increase in the target distance. It can mainly be attributed to the target small features the further its distance is. Figure 5.5b presents similar decline in precision, although not as sharp. Also, there are stark discrepancies between the precision on the hover and ground manoeuvres. These disparities are indicative of a large number of false positives on the hover experiments. There are two main phenomena at play that may be the root cause: disparities between the time the target is under the horizon and extra ground clutter captured while the system is in hover.

First, between the *ground* and *hover* experiments, the target spends a disproportional amount of time above or under the horizon. On the *ground*, most of the target trajectories happen above the sensor system, where the target spends about 84.4% of its trajectories above the horizon. In contrast, in the *hover* experiments this value drops to an average of 62.8%. These discrepancies may relate to the lower overall recall on the *hover* experiment, as the complex background under the horizon makes the detection task more difficult.

A second plausible explanation for the experiment discrepancies is that the sensors' higher altitude provides a more detailed top-down view on ground clutter. This clutter, in turn, gets misdetected as a drone. An example of such occurrences is the safety cones positioned on the test field, which are more

visible in the hover manoeuvres. They contribute to the lower precision, as the YOLOv3 can incorrectly misrepresent them as drones, as exemplified in Figure 5.4.

Table 5.3: Results of the *YOLO-based tracker* on the *free-flight* experiments.

Total Targets	True Positives	False Positives	Recall	Precision	MOTP	MOTA
9,847	4,770	23,238	48.4%	17.0%	72.0%	-1.88

Table 5.3 presents the *YOLO-Based tracker* results on the *free-flight* experiments, which show 48.4% target recall and a 17.0% precision on target capture. This experiment has, on average, 47.0% of the target trajectories spent under the horizon. Compared to 84.4% and 62.8% of frames spent above the horizon in the previous experiments, a relationship between recall values and the percentage of target frames spent above the horizon appears to exist. The analysis of each *free-flight* trajectory, presented in Figure 5.6, indicates a connection between the percentage of a successfully tracked trajectory and the amount of time that vehicle spends above the horizon. These results show that the tracking results depend on the object location, which is an important factor.

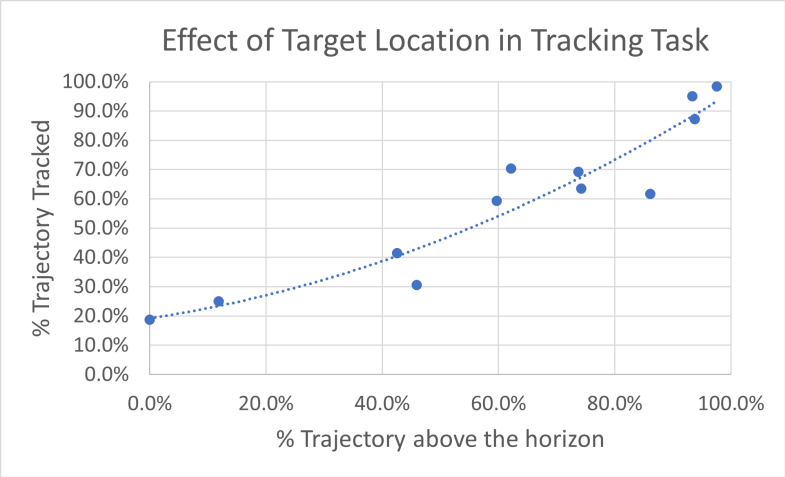


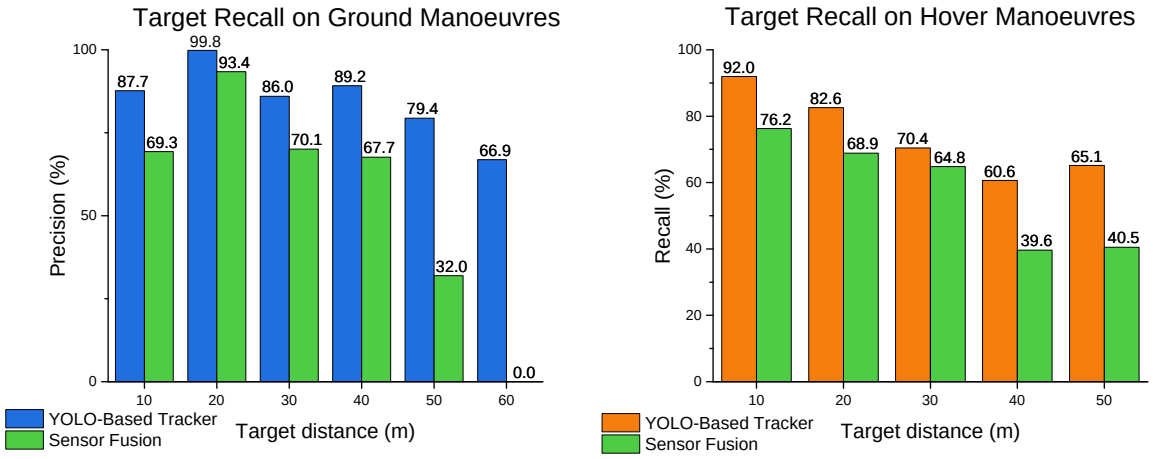
Figure 5.6: Analysis of trajectories from the *free-flight* experiment. There are in total 11 trajectories, being classified as follows: mostly tracker (MT) - 3; partially tracked (PT) - 8; mostly lost (ML) - 1.

5.4 Sensor Fusion Results

This section presents and compares the results of the sensor fusion algorithm. The *YOLO-based tracker* is kept with the same confidence and IOU threshold values as chosen in Section 5.3.1.

Figure 5.7 shows that the sensor fusion recall on both *ground* and *hover* experiments is comparable to the *YOLO-based tracker* on distances up to $30m$, with the sensor fusion presenting the lower values. One cause for this effect is the sensor fusion's idleness while waiting for the LiDAR sensor to capture a target, which leaves initial frames without detections. The increase of target distance amplifies this

effect, as a significant decline in recall values can be observed on distances over $40m$. Even with a LiDAR recall of 0.6% for the $50m$ range, the sensor fusion solution can still achieve a recall of 32.0% for the *ground* experiment. For the *hover* experiment, sensor fusion solution presents a recall of 40.5% despite the LiDAR sensor only capturing the target 1.2% of the time. The $60m$ distance did not have any detections in the sensor fusion method. It is understandable, as there was only a single LiDAR detection at this distance, and the calibration procedure did not project it correctly.

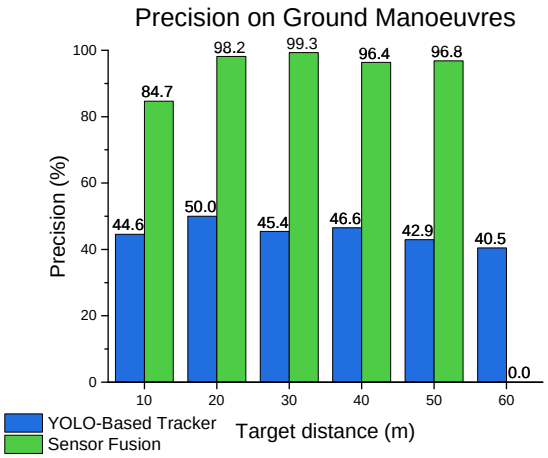


(a) Recall comparison on the *ground* experiments.

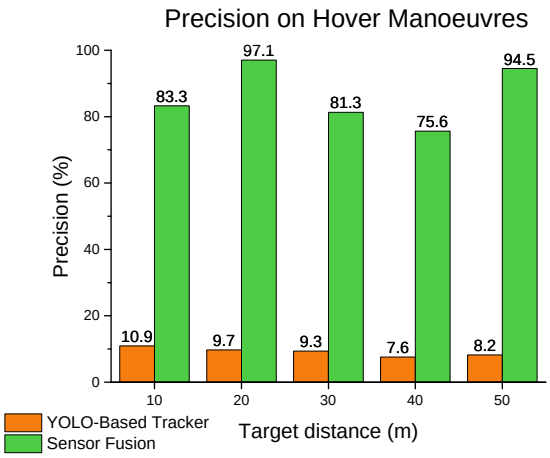
(b) Recall comparison on the *hover* experiments.

Figure 5.7: Recall comparison between sensor fusion and *YOLO-based tracker* on the *moving target* experiments.

Precision results are shown is in Figure 5.8. The sensor fusion solution shows a tremendous increase in precision, either on the *ground* or in the *hover* experiments. This effect is related to the region of interest (ROI) being focused on the target location, leading to the algorithm being less susceptible to false positives. *Ground* experiments show a consistent precision above 96%, except for the $10m$ range, which presents an 84.7% precision. In the *hover* experiments, there is a decrease in precision between $30m$ and $40m$, which rises to 94.5% at $50m$. This anomaly might be influenced by the presence of ground clutter on the *hover* experiments. Despite the sensor fusion solution presenting lower values of precision in the *hover* experiments, it is there that the most significant disparities with the *YOLO-based tracker* happen, having the sensor fusion almost ten times higher precision across target distances.



(a) Precision comparison on the *ground* experiments.



(b) Precision comparison on the *hover* experiments.

Figure 5.8: Precision comparison between sensor fusion and *YOLO-based tracker*.

Similar trends are shown for the *free-flight* experiments, seen in Table 5.4. The sensor fusion and the *YOLO-based tracker* present similar recall values, of 48.4% and 41.9%, respectively. Once again, the sensor fusion solution shows a drastic increase in target precision, from 17.0% to 91.2%. Both solutions present similar $MOTP$, which means the average IOU overlap between detections and ground truth is above 70% for both. Sensor fusion presents an increase from -1.88 to 0.38 in the MOTA metric, which accounts for the overall tracking ability. Although not referenced in Table 5.4, no identity switch ($IDSW$) were observed during the tracking process.

However, the sensor fusion solution presents a lower ability to reconstruct the ground truth track, dropping by one the number of trajectories *mostly tracked* (MT) and increasing by four the number of trajectories *mostly lost* (ML). A reason for this effect can be linked to some tracks being made of a lower number of frames, giving a smaller change for the LiDAR to detect the vehicle and begin the tracking procedure. Each time a RPA enters and leaves the camera field of view, a new trajectory is created, leading to a high variability in the length of each one. Nonetheless, the sensor fusion method presents half of the trajectory fragments, 50 instead of 99, showing better tracking consistency than the *YOLO-based tracker*.

Additionally, both solutions show the ability to process sensor data in real-time, presenting a processing speed higher than $20Hz$, which is the framerate at which sensor data is recorded. Furthermore, the sensor fusion methods can more than double the processing speed of the visual algorithm, to $57Hz$. However, these results are achieved on powerful computational resources, using an *NVIDIA Tesla P100 GPU* to improve the processing speed of the YOLOv3 detector. Real-time onboard capabilities is still a topic left to be researched.

Table 5.4: Tracking results of the *YOLO-based tracker* and sensor fusion on the *free-flight* experiments.

	Recall	Precision	MOTP	MOTA	MT	PT	ML	FM	Runtime
YOLO-Based Tracker	48.4%	17.0%	72.0%	-1.88	3	8	1	99	24 Hz
Sensor Fusion	41.9%	91.2%	74.0%	0.38	2	5	5	50	57 Hz

5.5 Discussion

The filtering methods explored were able to successfully removed non-target detections from the LiDAR's point cloud. The boundaries created were simple since the flight test environment was mostly clutter-free. A more complex scenario, such as an urban environment, might require more elaborate boundary formulations to remove such points.

The experimental tests indicate that LiDAR detections decay rapidly with the increase in target distance. These results are expected due to the sparse point cloud created by the M8 LiDAR, presenting only eight vertical channels. These results demonstrate the sensor's inability to detect a flying target across multiple ranges continuously. Thus, a complementary sensor is required if a reliable system is to be created. Additionally, a maximum LiDAR detection range of $60m$ was observed for the specific sUAS tested, the DJI Inspire, and might differ for other aircraft models. A study on diversified agents might be necessary to construct a reliable C-sUAS solution to deal with different threats.

When comparing the LiDAR performance between the *ground* and *hover* experiments, no large differences are presented. In theory, the hover position would provide a better detection situation since most of the horizontal scans would be looking for targets instead of pointing towards the ground. The similarity of results can also be attributed to the cross manoeuvre's adaptation to each, since the manoeuvre's maximum height, h_{max}, was dependant on the system's altitude, h_s, presenting larger altitude deviations in the *hover* experiment. These results also indicate the perturbations caused by the flying vehicle do not jeopardize the LiDAR detection ability.

Furthermore, the *cross manoeuvre*, in which the LiDAR detection performance was evaluated, represents an ideal scenario intending to increase LiDAR detections and might not reflect realistic flight conditions. Nonetheless, the results obtained provided an excellent data source for evaluating the detection and tracking performance across different target distances.

Regarding the *moving sensor* experiments, the *pitch angle manoeuvre* is more advantageous than the *altitude variation*, as the manoeuvre presents a higher recall rate and requires less energy to operate, as opposed to climbing various meters in the air. Also, with the *pitch variation* motion, the regions covered by each horizontal scan overlap each other for all ranges.

The extrinsic calibration results show a much better performance on the visual inspection approach than using a calibration board. The main reason for the calibration board's failure was the distances at which points were captured. A small error in calibration at close distances can translate into large deviations at longer ranges. A different approach able to estimate the 3D corner points more accurately could

improve results. Although more accurate, the visual inspection procedure is not an efficient methodology to perform each time a LiDAR and a camera need to be calibrated.

Additionally, the results obtained by the YOLOv3 detector are highly dependent on the target's position relative to the environment scene. These limitations are mainly caused by the training dataset, which did not prepare the detector to deal with textured backgrounds. Apparent false positives, such as the treetop, show noticeable flaws in the training process. However, this research constructed a labelled dataset without this setback, which can train the YOLOv3 detector to improve under the horizon results.

Despite the lower recall values shown in the sensor fusion results, compared to the *YOLO-based tracker*, it presented higher precision while tracking the target. This effect is observed across all experiments. This indicates that the *ROI creation* successfully focused the searching area of the visual detection algorithm, lowering the number of false positives acquired. The main reason for the lower recall in the sensor fusion method if the algorithm's idleness while waiting for a LiDAR detection since it relies entirely on the sensor for target acquisition. This effect is felt more intensely at larger distances where the point cloud is more sparse.

Chapter 6

Conclusions and Future Work

This chapter presents the main conclusion of this thesis and discusses possible steps for future work.

6.1 Conclusions

This thesis provides a multi-sensor methodology to detect and track sUAS. The approach dealt with non-cooperative targets through a combination of a LiDAR and an EO camera.

This research demonstrated two different methods to obtain calibration points to perform the extrinsic calibration procedure between a LiDAR and a camera. It showed that the combination of both methods would lead to the best accuracy results.

It also implemented a visual detection and tracking system by using a YOLOv3 detector and a modified DeepSORT tracker. This thesis presented an approach to deal with the computational setback associated with the visual target search and the subsequent capture of false positives. It successfully minimized these problems by using LiDAR detections to acquire the target location and creating a region of interest (ROI) around it.

This thesis has designed and constructed a sensor system payload, which was experimentally tested onboard an aircraft. Multiple flight test experiments were planned and executed by a team of specialists from the Centre for Aerospace Research (CfAR) of the University of Victoria. From these experiments, this research produced a multi-drone dataset with targets located in complex backgrounds.

6.2 Future Work

Potential future work includes the improvement of the YOLOv3's detection performance. The solution would include the use of a more appropriate dataset to train the algorithm. Ideally, it would involve various aircraft types to allow the algorithm to identify specific sUAS models and be located on textured backgrounds to make the detector more resilient to vehicles under the horizon. Additionally, the development of a dynamic ROI creation that adjusts according to the vehicle's appearance and location could present improvements in the approach's runtime and precision. Moreover, incorporating a visual search

on random image locations when the LiDAR does not return any targets and when no target is being visually tracked could also improve recall results.

Further future work could include creating an algorithm capable of locating a vehicle in a 3D environment, taking advantage of LiDAR measurements. This information would be precious for a C-sUAS system. Furthermore, algorithm optimization and subsequent evaluation on an onboard computer would be needed to prove its real-time onboard capabilities. Finally, creating a simulated environment to capture additional data and perform extra manoeuvres would be extremely valuable, considering flight tests are expensive and take a lot of time to prepare and execute.

Bibliography

[1] L. Swann. *Unmanned Aircraft Systems (UAS)*. International Civil Aviation Organization, 2011. ISBN 9789292317515.

[2] F. Nex and F. Remondino. Uav for 3d mapping applications : a review. *Applied geomatics*, 6(1): 1–1–2015, 2014. ISSN 1866-9298. doi: 10.1007/s12518-013-0120-x.

[3] G. J. Adabo. Long range unmanned aircraft system for power line inspection of brazilian electrical system. *Journal of Energy and Power Engineering*, 8(2), 2014.

[4] M. N. Gillins, D. T. Gillins, and C. Parrish. Cost-effective bridge safety inspections using unmanned aircraft systems (uas). In *Geotechnical and Structural Engineering Congress 2016*, pages 1931–1940, 2016.

[5] D. P. Thipphavong, R. Apaza, B. Barmore, V. Battiste, B. Burian, Q. Dao, M. Feary, S. Go, K. H. Goodrich, J. Homola, et al. Urban air mobility airspace integration concepts and considerations. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3676, 2018.

[6] R. Schaufele and M. Lukacs. FAA Aerospace Forecasts: Fiscal Years 2020-2040. pages 1–119, 2019. `https://www.faa.gov/data_research/aviation/aerospace_forecasts/` [Online: accessed on December 2020].

[7] The New York Times. White House Drone Crash Described as a U.S. Worker's Drunken Lark, 2015. `https://www.nytimes.com/2015/01/28/us/white-house-drone.html` [Online: accessed on January 2021].

[8] The Guardian. Gatwick drone disruption cost airport just £1.4m, 2018. `https://www.theguardian.com/uk-news/2019/jun/18/gatwick-drone-disruption-cost-airport-just-14m` [Online: accessed on December 2020].

[9] UK Airprox Board. Airprox Reports 2020, 2020. `https://www.airproxboard.org.uk/Reports-and-analysis/Airprox-Reports-2020/` [Online: accessed on December 2020].

[10] The Age. Prisons struggle to swat drug-smuggling drones. `https://www.theage.com.au/national/victoria/prisons-struggle-to-swat-drug-smuggling-drones-20201115-p56ep1.html` [Online: accessed on December 2020].

[11] Transport Canada. Flying your drone safely and legally, 2020. https://tc.canada.ca/en/aviation/drone-safety/flying-your-drone-safely-legally [Online: accessed on December 2020].

[12] BBC News. Venezuela President Maduro survives 'drone assassination attempt', 2018. https://www.bbc.com/news/world-latin-america-45073385 [Online: accessed on December 2020].

[13] NATO Industrial advisory group. Engagement of low, slow and small aerial targets by GBAD. *final report on NATO NIAG study group*, 2013.

[14] NATO Industrial advisory group. GBAD sensor mix optimization. *final report on NATO NIAG study group 188*, 2015.

[15] NATO Industrial advisory group. Low, slow and small threat effectors. *final report on NATO NIAG study group 200*, 2017.

[16] B. Wilson, S. Tierney, B. Toland, R. M. Burns, C. P. Steiner, C. S. Adams, M. Nixon, R. Khan, M. D. Ziegler, J. Osburg, and I. Chang. *Small Unmanned Aerial System Adversary Capabilities*. RAND Corporation, Santa Monica, CA, 2020. doi: 10.7249/RR3023.

[17] A. Silkoset, A. Holland, M. Bruno, and O. Martins. *Countering the Drone Threat Implications of C-UAS technology for Norway in an EU and NATO context*. Peace Research Institute Oslo (PRIO), 2020. ISBN 978-82-343-0074-5.

[18] K. Wackwitz. Counter-Drone Market Report 2020, 2019. https://droneii.com/counter-drone-market-report-2020 [Online: accessed on December 2020].

[19] R. V. Simonsen, M. Hartung, K. C. Brejndal-Hansen, S. Yding Sørensen, K. O. Sylvester-Hvid, and D. Klein. Global Trends of Unmanned Aerial Systems. *Danish Technological Institute [DTI]*, page 44, 2019. https://www.auvsi.org/global-trends-unmanned-aerial-systems [Online: accessed on December 2020].

[20] P. Andraši, T. Radišić, M. Muštra, and J. Ivošević. Night-time detection of uavs using thermal infrared camera. *Transportation Research Procedia*, 28:183 – 190, 2017. ISSN 2352-1465. doi: 10.1016/j.trpro.2017.12.184. INAIR 2017.

[21] S. Dogru and L. Marques. Pursuing drones with drones using millimeter wave radar. *IEEE Robotics and Automation Letters*, 5(3):4156–4163, July 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.2990605.

[22] D. L. Hall and S. A. McMullen. *Mathematical techniques in multisensor data fusion*. Artech House, 2004.

[23] A. H. Michel. *Counter-Drone Systems: 2nd Edition*. Center for the Study of the Drone at Bard College, December 2019. https://dronecenter.bard.edu/projects/counter-drone-systems-project/ [Online: accessed on December 2020].

[24] M. Hammer, B. Borgmann, M. Hebel, and M. Arens. A multi-sensorial approach for the protection of operational vehicles by detection and classification of small flying objects. In *Electro-Optical Remote Sensing XIV*, volume 11538, page 1153807. International Society for Optics and Photonics, 2020.

[25] M. Hammer, M. Hebel, M. Laurenzis, and M. Arens. Lidar-based detection and tracking of small UAVs. In G. S. Buller, R. C. Hollins, R. A. Lamb, and M. Mueller, editors, *Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures*, volume 10799, pages 177 – 185. International Society for Optics and Photonics, SPIE, 2018. doi: 10.1117/12.2325702.

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6): 1137–1149, 2016.

[27] R. Opromolla, G. Inchingolo, and G. Fasano. Airborne visual detection and tracking of cooperative uavs exploiting deep learning. *Sensors*, 19(19):4332, 2019.

[28] J. Li, D. H. Ye, T. Chung, M. Kolsch, J. Wachs, and C. Bouman. Multi-target detection and tracking from a single camera in unmanned aerial vehicles (uavs). In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4992–4997. IEEE, 2016.

[29] D. H. Ye, J. Li, Q. Chen, J. Wachs, and C. Bouman. Deep learning for moving object detection and tracking from a single camera in unmanned aerial vehicles (uavs). *Electronic Imaging*, 2018(10): 466–1, 2018.

[30] M. Vrba, D. Heřt, and M. Saska. Onboard marker-less detection and localization of non-cooperating drones for their safe interception by an autonomous aerial system. *IEEE Robotics and Automation Letters*, 4(4):3402–3409, 2019.

[31] M. Vrba and M. Saska. Marker-less micro aerial vehicle detection and localization using convolutional neural networks. *IEEE Robotics and Automation Letters*, 5(2):2459–2466, 2020.

[32] M. U. de Haag, C. G. Bartone, and M. S. Braasch. Flight-test evaluation of small form-factor lidar and radar sensors for suas detect-and-avoid applications. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–11. IEEE, 2016.

[33] D. Justino, R. Ventura, and A. Suleman. Lidar and camera sensor fusion for air-to-air detection and tracking of airborne intruders. Unmanned Systems Canada Conference, November 3rd 2020.

[34] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1(February), 2001. ISSN 10636919. doi: 10.1109/cvpr.2001.990517.

[35] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[36] OPENGENUS. Using histogram of oriented gradients (HOG) for object detection. `https://iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/` [Online: accessed on December 2020].

[37] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[39] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[40] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[41] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[42] T. Sen, M. K. Hasan, M. Tran, Y. Yang, and M. E. Hoque. Selective Search for Object Recognition. *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, pages 357–364, 2018. doi: 10.1109/FG.2018.00058.

[43] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[44] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[45] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[46] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[47] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[48] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[49] Y. Yang and H. Deng. Gc-yolov3: You only look once with global context block. *Electronics*, 9(8): 1235, 2020.

[50] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[51] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[52] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[53] C. Mallet and F. Bretar. Full-waveform topographic lidar: State-of-the-art. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(1):1 – 16, 2009. ISSN 0924-2716. doi: 10.1016/j.isprsjprs.2008.09.007.

[54] J. Shan and C. K. Toth. *Topographic laser ranging and scanning: principles and processing*. CRC press, 2018.

[55] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997. doi: 10.1109/CVPR.1997.609468.

[56] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[57] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6): 381–395, 1981.

[58] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[59] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[60] F. Svanström. Drone Detection and Classification using Machine Learning and Sensor Fusion. Master's thesis, Halmstad University, 2020.

[61] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[62] MATLAB. version 9.6.0.1072779 (R2019a), 2019.

[63] Y. Park, S. Yun, C. S. Won, K. Cho, K. Um, and S. Sim. Calibration between color camera and 3d lidar instruments with a polygonal planar board. *Sensors*, 14(3):5333–5353, 2014.

[64] Q.-Y. Zhou, J. Park, and V. Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.

[65] M8 ™ Sensor User Guide, 2019. Available at `http://quanergy.com` [Online: accessed on December 2020].

[66] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford. Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system. *Electronics*, 7(6):84, 2018.

[67] Stanford Artificial Intelligence Laboratory et al. Robotic operating system, 2016. `https://www.ros.org` [Online: accessed on December 2020].