Deep learning based drone localization and payload detection using vision data

by

Hamid Azad

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Applied Science
in
Electrical and Computer Engineering with Concentration in Applied Artificial Intelligence

Examining Committee

The following served on the Examining Committee for this thesis.

Carleton Member:	Dr. Richard M. Dansereau Professor, Department of Systems and Computer Engineering Carleton Univeristy
Internal Member:	Dr. Martin Bouchard Professor, School of Electrical Engineering and Computer Science University of Ottawa
Supervisor:	Dr. Miodrag Bolic Professor, School of Electrical Engineering and Computer Science University of Ottawa
Co-Supervisor:	Dr. Iraj Mantegh Senior Research Officer and Technology Lead, Aerospace Research Centre National Research Council Canada, Montreal, Canada

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Uncrewed aerial vehicles (UAVs), commonly known as drones, have become increasingly prevalent in various applications. However, the localization and payload detection of drones is crucial for ensuring safety and security. This thesis proposes a vision-based solution using deep learning techniques to address these challenges.

Existing solutions like radars and acoustic sensors have limitations, including high costs, limited accuracy, and the need for prior knowledge of the drone's model. Normal radars lack angle estimation accuracy and rely on known micro-Doppler features for payload detection, while acoustic sensors are restricted to close ranges for payload analysis. In contrast, cameras offer a cost-effective alternative as they have become widely available and can capture visual data. In addition, due to resource constraints, mounting multiple sensors on the UAV along with the camera is impractical, making reliance on cameras alone essential for addressing the mentioned problems. Recent advancements in deep learning algorithms enable regression and classification tasks, making vision data a promising choice for solving drone localization and payload detection problems.

The proposed solution leverages convolutional neural networks (CNNs) for regression tasks, estimating the distance of a drone from the captured image. The CNN takes a cropped image within the drone's bounding box as input and outputs the estimated distance. Additionally, the drone's azimuth and elevation angles have been estimated based on its position in the captured image using a simple pinhole model for the camera. Also, the ResNet and EfficientNet classifiers could accurately classify drones as loaded or unloaded, even without prior knowledge of their shape. Due to a scarcity of publicly available datasets, this study developed the first air-to-air simulated dataset specifically for the classification of loaded versus unloaded drones.

To evaluate the performance of the proposed solution, both simulated and experimental tests were conducted. The results showcased promising accuracy, with a root mean square error (RMSE) of less than 10 meters for distance estimation and an RMSE of less than 3 degrees for angle estimation. Furthermore, the payload detection problem was effectively addressed, achieving a classification accuracy of over 85% for distinguishing between loaded and unloaded drones using the trained network based on the simulated dataset. The numerical highlights demonstrate the effectiveness of using camera sensors for 3D localization, with accurate distance and angle estimations. The high accuracy achieved in payload classification showcases the potential of the proposed solution for detecting drone payloads at distances up to 100 meters. These results pave the way for enhanced safety and security in drone environments.

Acknowledgements

I would like to express my sincere gratitude to my dedicated thesis supervisor, Professor Miodrag Bolic, and my esteemed co-supervisor, Dr. Iraj Mantegh, for their unwavering support and guidance throughout the completion of my master's thesis. Professor Bolic's profound knowledge, commitment, and mentorship have been instrumental in shaping this thesis. His insightful feedback and encouragement propelled me to explore new horizons in my research and approach challenges with confidence. I would also like to extend my heartfelt appreciation to Dr. Mantegh for his valuable input and guidance. His expertise and guidance enriched the quality of my work and added depth to my research. I am sincerely grateful for the trust and support that both Professor Bolic and Dr. Mantegh have provided, enabling me to reach this significant milestone in my academic journey. Their dedication to my success has been truly invaluable.

I would also like to acknowledge the unwavering support of my loving family. My wife, Shaghayegh, and my son, Ryan, have been my pillars of strength and a constant source of inspiration. Their patience, encouragement, and understanding during this journey have been my driving force. My heartfelt thanks also extend to my loving family, especially my father and mother, who reside in my home country far away. Their unwavering belief in me and their constant encouragement has been a driving force throughout my academic journey. Their support, despite the distance that separates us, has been a source of strength and motivation.

I am also thankful to the entire team in the Computational Analysis and Applications Research Group (CARG) lab at the University of Ottawa for their support and the conducive research environment they provided. This journey would not have been possible without their help.

Finally, I would like to thank National Research Council Canada (NRC) for funding this work through the *Artificial Intelligence for Logistics* project titled *"Beyond visual line of sight (BVLOS) inspection of critical infrastructures and superstructures using coordinated multiple unmanned aircraft systems (UAS)"*, and Natural Sciences and Engineering Research Council of Canada (NSERC) for partially supporting the project through *NSERC Discovery* project titled *"Machine Learning with Uncertainty for Monitoring Moving Objects and People"*.

Table of Contents

List of Tables

List of Figures

Abbreviations

AirSim Aerial Informatics and Robotics Simulation viii, x, xi, 6, 32, 33, 35, 36, 46, 52, 68, 70–72, 101, 102, 105, 107, 109, 112, 113, 123–126

BB Bounding Box xii–xvii, 4, 5, 20, 23, 31, 44–48, 70–73, 75, 76, 82, 83, 85–87, 89, 94–96, 107, 109, 124

CNN Convolutional Neural Network 2, 6, 12, 15, 19, 22, 23, 30, 59, 61, 71–73, 76, 84, 86, 107, 110

DL Deep Learning 12, 14–16, 20–22, 27, 28, 53, 54, 62, 70, 82, 84, 85, 99, 107, 109–111

DNN Deep Neural Network 22

EfficientNet EfficientNet ix, 51

EO Electro-Optical 10

FAA Federal Aviation Administration 1

FOV Field of View 16, 35, 42, 56, 57, 71

FPS Frames Per Second 17

GPS Global Positioning System xv–xvii, 23, 47, 55, 62, 63, 66, 68, 70, 76, 83, 87–89, 93, 95, 97, 99, 110

GT Ground-Truth xv–xvii, 68–70, 77, 78, 83, 84, 87, 88, 91–93, 95, 97–100, 102, 103

GUI Graphical User Interface viii, xvii, 7, 101, 105–107, 123–126

KITTI Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago 19, 21, 22

KNN K-Nearest Neighbor 15

MA Moving Average 58, 73, 110, 111

mAP mean Average Precision 17

MFCC Mel-frequency cepstral coefficients 5, 15, 16, 24

ML Machine Learning 15, 20, 27, 28, 106

NRC National Research Council of Canada xv, 42, 77, 80

PTZ Pan-tilt-zoom 3, 4, 62, 101–104

RCS Radar Cross Section 13, 112

ReLU Rectified Linear Unit 17, 29, 60, 61

ResNet Residual neural Network ix, x, xii, xiii, 2, 5, 17, 19, 20, 26–31, 39–41, 44–52, 109–111

RGB Red-Green-Blue 17, 21, 112

RMSE Root Mean Square Error 6, 73, 76, 84, 86, 87, 91, 94, 96–98, 100

RNN Recurrent Neural Network 15, 112

SORT Simple Online and Realtime Tracking 4

STFT Short-Time Fourier Transform 11, 12

SVD Singular Value Decomposition 11

SVM Support Vector Machine 13–16

TSS Total Station Surveying xiv, 69, 77–79

UAV Uncrewed Aerial Vehicle x, xiv, 1–7, 9, 10, 12, 14, 15, 17, 22–25, 32, 44, 53, 61, 62, 69, 77–80, 101, 108, 110

UCL dataset University College London collected dataset 10, 12, 24

VGG Visual Geometry Group 22

YOLO You Only Look Once 4, 5, 17, 20–23, 30, 53, 56, 57, 59, 70, 79, 81, 112

Chapter 1

Introduction

UAV, also known as drones, have attracted considerable attention recently [9–15]. The U.S. Federal Aviation Administration (FAA) released its first approval of UAVs to be integrated into the United States' airspace in November 2013 [16]. Due to FAA Aerospace Forecast 2022-2042 [17], the number of registered UAVs with FAA by December 2020 was about 1.37 million (with an average of 10300 per month during 2021). In Canada, this number was over 72,000 by December 2022, according to Transport Canada. Regarding their relatively small size and the ability to fly and be guided by remote control, utilizing drones have grown dramatically in both commercial and governmental tasks such as:

- Agriculture
- Cargo transport and home delivery
- City surveillance
- Search and rescue activities in disasters
- Traffic monitoring
- Aerial photography
- Fire fighting
- Border security

However, the vast usage of UAVs raises serious concerns about safety and security issues. Threats such as transporting smuggled goods and other illegal substances, illegal entry to restricted areas (no-fly zones), conducting physical and cyber attacks by terrorists, and also accidents with planes in airports are among many concerns about UAVs. Thus, the counter-UAV measures, including detection, localization, and classification of UAVs, have received a steady increase in attention during these years. For instance, searching with the keywords "drone detection" as the first step to counter-UAV measures shows a rapid increase in the number of research papers in the recent ten years in IEEE database content (Figure 1.1) [18].

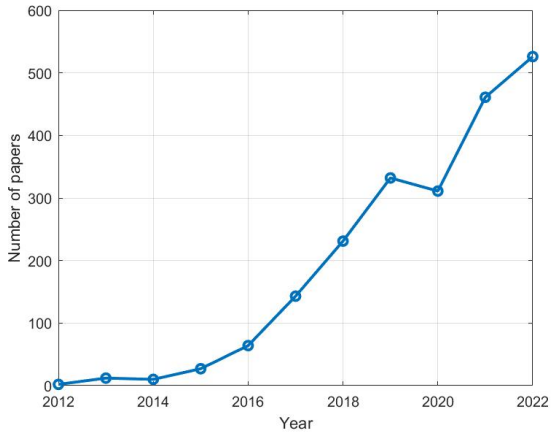


Figure 1.1: IEEE database content (number of papers) for "drone detection" from 2012 to 2022

Once the target drone is detected, the two primary challenges are accurately determining its location and determining whether it is loaded or unloaded. This thesis presents a vision-based solution using deep learning techniques to tackle the challenges of drone localization and payload detection. By utilizing cameras as a cost-effective sensor, the proposed solution utilizes Convolutional Neural Network (CNN) for regression tasks to estimate drone distance. Moreover, the ResNet and EfficientNet classifiers are utilized to classify between loaded and unloaded drones without prior shape knowledge. To address the lack of available datasets, an air-to-air simulated dataset specifically for loaded versus unloaded drone classification is developed. The proposed approaches are also implemented and validated on both simulated and experimental datasets.

1.1 Motivation and problem statement

Consider the multi-sensor counter-UAV system depicted in Figure 1.2. This system comprises a radar and a Pan-tilt-zoom (PTZ) camera mounted on the ground, serving as early warning sensors for identifying target drones. However, these ground-based platforms have limitations in terms of flexibility and coverage area for long distances, which may hinder their effectiveness in mitigating potential threats posed by the target drone. To address these limitations, an alternative approach involves equipping a counter-UAV sensor on an observer drone, enabling it to actively pursue and neutralize the target drone. By utilizing the mobility of the observer drone, the defense coverage area can be expanded, thereby enhancing the system's effectiveness against the target drone. Nevertheless, it is not feasible to mount multiple sensors due to constraints such as payload size, weight, and limited power supply on the observer drone. In such cases, a vision camera emerges as a cost-effective solution. Vision-based sensing offers several advantages, including affordability, compact implementation, and lower power consumption. Thus, as illustrated in Figure 1.2, the camera becomes the preferred choice for the counter-UAV sensor mounted on the observer drone.

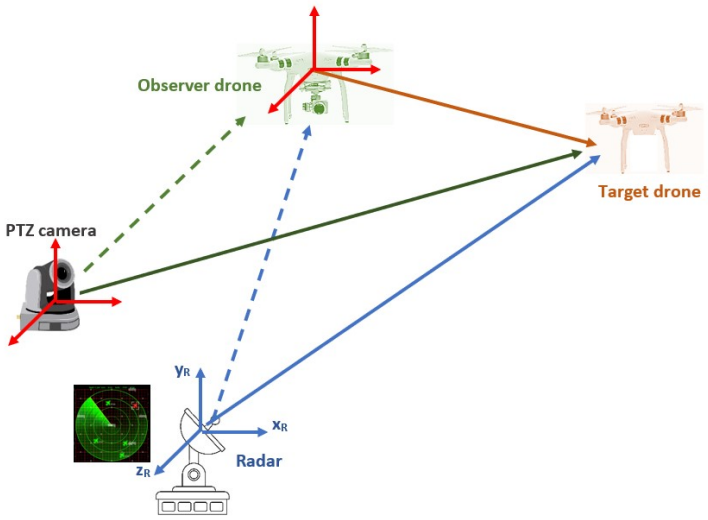


Figure 1.2: Multi-sensor counter-UAV system

Under these circumstances, the only available counter-UAV sensor onboard would be the vision camera. The captured images and videos from this sensor play a vital role in performing the identification task for subsequent neutralization, extracting crucial infor-

mation about the target drone such as its location and whether it is carrying a payload or not. Localization of the target drone enables the determination of appropriate defense measures based on its proximity to the restricted area. Furthermore, payload detection plays a significant role in assessing the potential threat posed by the target drone. This capability enhances overall situational awareness and enables appropriate decision-making regarding the level of response required.

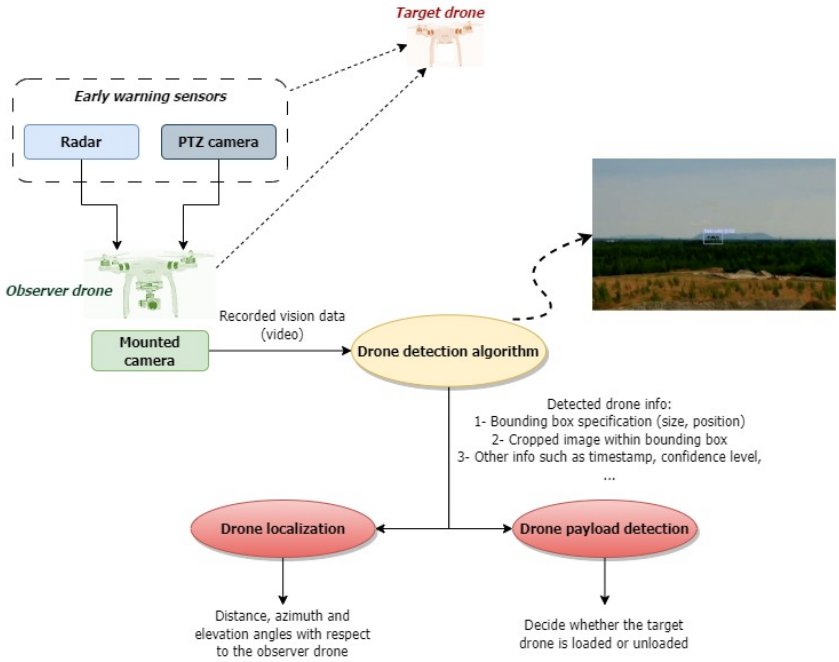


Figure 1.3: Multi-sensor counter-UAV system

The block diagram corresponding to the multi-sensor counter-UAV system (Figure 1.2) is illustrated in Figure 1.3. Once the target drone is detected using the radar and PTZ camera, a command is issued to the observer drone to approach the approximate location of the target for closer inspection. The video captured by the onboard camera on the observer drone is continuously processed using an object detection algorithm [19–21]. The detection algorithm is beyond the scope of this thesis, but it is assumed that the You Only Look Once (YOLO)-v7 algorithm [22–24] specifically trained for drone detection, or its combination with Simple Online and Realtime Tracking (SORT) with a deep association metric (DeepSORT) tracker [25, 26] will be applicable. These algorithms provide valuable outputs for further analysis, including the position and size of the BB (in pixels), the timestamp of detection, and the confidence level of being a drone for the detected object.

The extracted BB information allows for cropping the region of interest from the image frame, which is then passed to the subsequent steps. The localization and payload detection algorithms are then applied to the detected drone. The focus of this study will be on these two blocks, highlighted in red color in Figure 1.3, and the corresponding algorithms.

It is worth mentioning that regarding localization and payload detection, existing solutions like radars and acoustic sensors (mostly mounted on the ground) have limitations including high costs, limited accuracy, and the need for prior knowledge of the drone's model. Normal radar lacks accuracy for angle estimation. Additionally, it relies on prior known micro-Doppler features to determine drone payload status. Acoustic sensors analyze signals using Mel-frequency cepstral coefficients (MFCC) but require knowledge of normal (unloaded) scenario features. Besides, sound signals are limited to close ranges (less than approximately 20 meters) for payload detection.

1.2 Objectives and contributions

Based on the aforementioned details in this thesis, we focus on tackling two key challenges that arise in the context of the problem described above. Firstly, utilizing the camera installed on the observer drone, our objective is to derive 3D location information of the target drone from the captured 2D image data. Secondly, leveraging the recorded vision data, we aim to determine whether the target drone is carrying a payload or not. Additionally, given the multi-sensor nature of the counter-UAV system, we also develop a preprocessing algorithm to align the measurements from different sensors, ensuring consistent and reliable data fusion.

The entire framework of this study is illustrated in Figure 1.4, depicting the research problems and their effects, the implications, and finally, the results achieved through this research work. Based on this framework, and assuming the preliminary detection of the target drone using an object detection method such as YOLO, the key contributions of this thesis can be summarized as follows:

- Binary classification of Loaded vs. Unloaded Drones: Various ResNet and EfficientNet classifiers are employed to distinguish between loaded and unloaded drones. Given the scarcity of publicly available datasets for this problem, we develop the first air-to-air simulated dataset, encompassing diverse drone types, payload variations, backgrounds, and weather conditions. The trained network demonstrates its effectiveness, achieving a classification accuracy exceeding 85% when addressing the

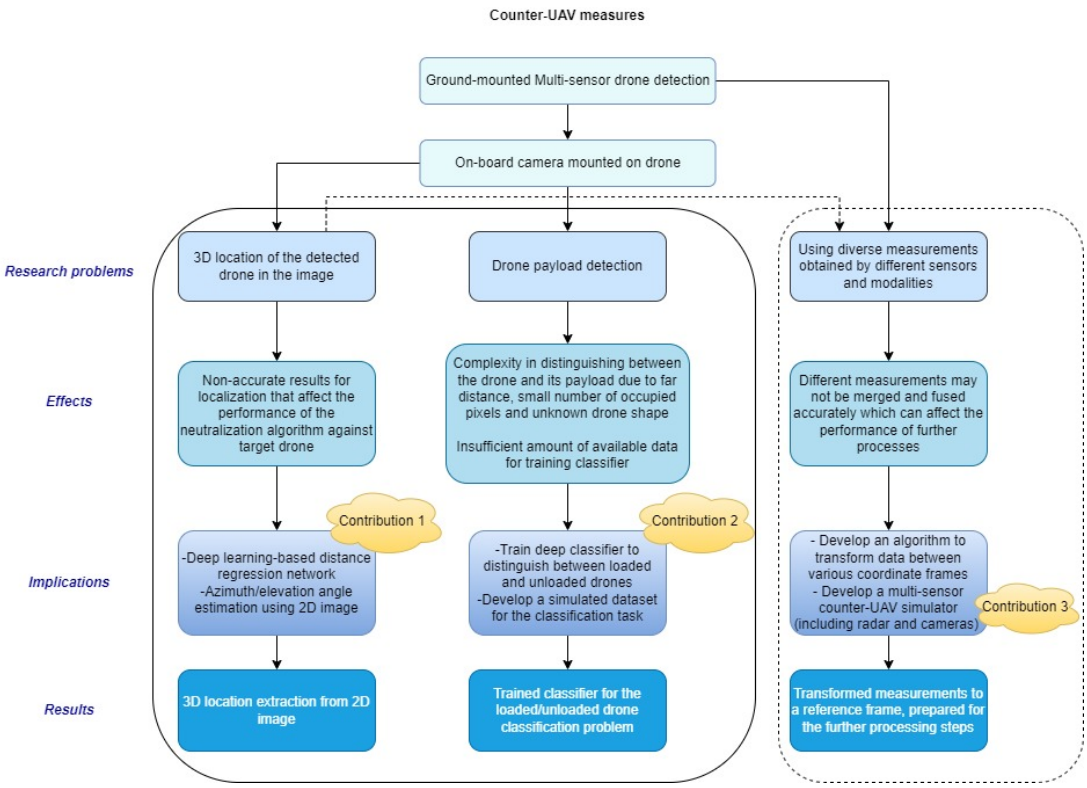


Figure 1.4: Research problems and their effects on the performance of the system, as well as the contributions of the thesis.

loaded vs. unloaded classification problem. Performance analysis is conducted using both simulated and experimental data.

- Distance and Angle Estimation: A deep learning-based method is developed to estimate the distance of the target drone from the camera using a CNN-based regression network. Additionally, azimuth and elevation angles are estimated using the camera's pinhole model to provide comprehensive information for 3D localization. The algorithm's performance is assessed using the simulated dataset and experimental data, achieving an Root Mean Square Error (RMSE) of less than 10 meters for distance estimation and 3 degrees for azimuth/elevation angles. Smoothing and tracking filters are also applied to the estimated distance trajectory to minimize errors and enhance the results.

- Multi-sensor counter-UAV simulator: A multi-sensor simulator using AirSim© in-

cluding both radar and vision camera sensors has been developed in this thesis. The simulator is designed as a Python-based GUI that allows the user to define drones' flight paths, interactively. The GUI offers customization options for various flight parameters, including drone model and altitude, as well as radar parameters such as its coverage angles in azimuth and elevation directions, and its position. The simulator provides synchronized multi-sensor simulated data, including radar and camera sensors, facilitating sensor fusion applications.

This study has resulted in the following research papers (and one journal paper which is under preparation):

- H. Azad, V. Mehta, M. Bolic, and I. Mantegh, "Simulated Dataset for the Loaded vs. Unloaded UAV Classification Problem Using Deep Learning", 2023 IEEE Sensors Applications Symposium (SAS), Ottawa, ON, Canada, 2023, pp. 1-6.
- H. Azad, V. Mehta, F. Dadboud, M. Bolic, and I. Mantegh, "Air-to-Air Simulated Drone Dataset for AI-powered problems", in Proc. of 42nd AIAA/IEEE Digital Avionics Systems Conference (DASC 2023)-*Selected for the Best of Poster Session Award*.
- V. Mehta, H. Azad, F. Dadboud, M. Bolic, and I. Mantegh, "Real-Time UAV and Payload Detection and Classification System Using Radar and Camera Sensor Fusion", in Proc. of 42nd AIAA/IEEE Digital Avionics Systems Conference (DASC 2023)-*Selected for the Best of Session Award*.

1.3 Thesis outline

The thesis comprises five chapters, each addressing different aspects of the study. In the first chapter, the overall block diagram of the counter-UAV system is presented, providing an overview of the research motivation and the problem under investigation. The objectives and contributions of the study are also outlined and discussed. In Chapter 2, a comprehensive review of the state-of-the-art in techniques relevant to the two main problems of this study, namely drone payload detection and localization, is presented. This chapter examines existing methodologies, analyzes their strengths and weaknesses, and identifies research gaps. Chapter 3 delves into the structure of the selected classifier, discussing its architecture and relevant details. Furthermore, the development of an air-to-air simulated dataset specifically designed for the loaded vs. unloaded classification

problem is described. The performance of the classifier is thoroughly analyzed using both simulated and experimental data. Chapter 4 focuses on the development of an algorithm for extracting the 3D location of the target drone using vision data. The methodology is described in detail, and the performance of the proposed algorithm is evaluated through various performance curves and plots, using both simulated and experimental datasets. The final chapter provides a comprehensive conclusion to the entire study. It summarizes the findings and highlights the advantages and limitations of the proposed approaches. Additionally, potential future research directions and areas for improvement are discussed.

Chapter 2

Literature review

2.1 Introduction

This chapter presents a comprehensive literature review on two significant aspects of drone technology: drone payload detection and drone localization using vision data. Drones, also known as UAVs, have gained immense popularity and utility in various fields, including surveillance, delivery services, and aerial photography. Accurate localization of drones and the ability to detect their payloads are crucial for ensuring their safe and efficient operation.

2.2 Drone Payload Detection

Payload detection involves the identification of objects or substances carried by drones [27]. This capability is crucial for security, law enforcement, and safety purposes, as it allows authorities to detect and prevent the transportation of illicit or dangerous payloads. Vision-based payload detection methods leverage visual data to analyze and interpret the target drones and determine the presence or absence of any probable payload. This section presents a comprehensive review of the existing literature on drone payload detection. The review covers approaches based on using various sensor modalities and the pros and cons of each. Drone payload detection has diverse practical applications across various domains [28]:

- **Security and Law Enforcement**: Accurate payload detection enables security personnel and law enforcement agencies to identify drones that may pose threats

to public safety, critical infrastructure, or sensitive areas (such as those carrying illegal substances, weapons, or contraband). By distinguishing between loaded and unloaded drones, security measures can be implemented to mitigate potential risks effectively.

- **Airspace Monitoring and Control**: Drone payload detection plays a crucial role in monitoring and controlling airspace activities. Identifying payload-carrying drones helps detect unauthorized or suspicious drones operating in restricted areas, ensuring compliance with airspace regulations and enhancing overall airspace security.
- **Border Control and Customs**: The detection of drone payloads is of paramount importance in border control and customs operations. It enables authorities to identify drones involved in smuggling activities, trafficking, or transporting illicit goods across borders. Effective payload detection enhances border security measures and contributes to combating cross-border crimes.

Various sensors, such as radar, Electro-Optical (EO) sensors, and acoustic sensors, have been employed for drone detection purposes [29–31]. These sensors can also be utilized for the classification of drones as either loaded or unloaded. Previous studies in the literature predominantly focus on classification methods that utilize recorded radar data from drones. A majority of those considered the micro-Doppler signature from the UAV and extracted some discriminating features to classify between loaded and unloaded drones [1, 2, 32–38]. However, there are also some works done using other sensors. In the following lines, various classification techniques are reviewed. These methods are categorized based on the employed sensor.

2.2.1 Radar

As the first attempt to use radar data for the problem of drone payload detection, the authors are [32] focused on analyzing the micro-Doppler signatures of micro-drones carrying different payloads and investigating the effectiveness of features for classifying and distinguishing between these cases. The study aimed to address the challenge of detecting micro-drones through radar and assess the potential of using multistatic radar instead of conventional monostatic radar for improved classification. Experimental data for this study were collected using the NetRAD [1] multistatic radar system at University College London[2]. The system consists of three identical nodes operating at 2.4 GHz. Two

[1] Netted RADar

[2] From now on, this dataset will be called University College London collected dataset (UCL dataset)

of them functioned solely as receiver nodes, while the third one, positioned between the other two nodes, served as a transmitter/receiver node. The experiment involved linear up-chirp modulation with 45 MHz bandwidth, allowing the micro-Doppler signature of the micro-drone to be captured.

The micro-Doppler signatures were analyzed through Short-Time Fourier Transform (STFT) to extract features. The experiment took place in July 2015 at the UCL Sports Ground, where the three NetRAD nodes were deployed with a 50 m inter-node separation. The micro-drone used was a DJI Phantom Vision 2+ without a camera but carrying different payloads weighing 10 g each. Three datasets were recorded for no payload, 200 g, and 500 g payload. Two features based on the Doppler and bandwidth centroid of the micro-Doppler signatures were identified as suitable for the loaded/unloaded classification. The classifiers employed were Naïve Bayes and diagonal-linear discriminant analysis. The classification performance was evaluated by training the classifiers with a portion of the available data and calculating the classification error. The results demonstrated that using multistatic data and combining the outputs of separate classifiers improved the classification accuracy compared to monostatic data or a single classifier. The overall classification accuracy consistently exceeded 90%, with the binary voting approach achieving 100% accuracy in some cases.

The same research team expanded their work in [33] by considering the previously mentioned classifiers and features as well as the random forest classifier and new features. By extracting features from the centroid of multistatic micro-Doppler signatures, they achieved successful classification with high accuracy (above 96-97%) for their multi-class problem. Based on the weight of the payload, five classes of payload were introduced, including zero weight (unloaded case), 200 g, 300 g, 400 g, and 500 g. Among several features used in this study, it was shown that the time-domain features based on the radar's received power are not suitable, while those based on the Singular Value Decomposition (SVD) decomposition of spectrograms were meaningful and effective. Regarding the structure for applying classifiers, the authors showed that using separate classifiers for each radar node improved classification accuracy compared to a centralized approach. In other words, instead of collecting all the data in a central processing unit and applying the classifier to the resultant data, the data from each node should be passed through the classifier and then strategies such as binary voting should be employed. The study also presented above 95% accuracy in the classification of micro-drones flying with different payloads using SVD features that were extracted from one-second length of data. Different features were found to be suitable for different scenarios, such as SVD-based features performing better for flying micro-drones and centroid-based features for hovering micro-drones. This suggests the need for different types of features in different scenarios and paths, and the advantage

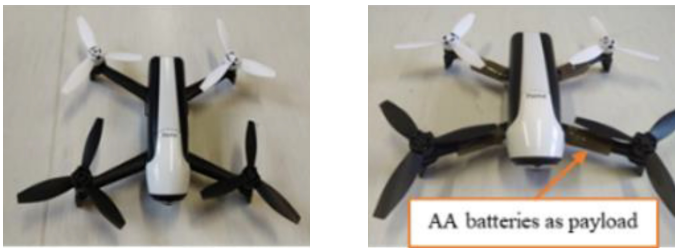
of utilizing different features from different radar nodes in a multistatic system to maximize classification accuracy.

In [34], the research team also analyzed the collected UCL dataset using Deep Learning (DL) approach. The data was processed offline by applying the Hilbert transform and matched filtering to increase the signal-to-noise ratio. A detector based on the constant false alarm rate (CFAR) principle was developed with the aim of detecting the range cells corresponding to the operation of the drone and determining whether it was in a flying or hovering state. Transformations such STFT, cepstrogram, and cyclic variation diagram (CVD) were applied to the identified range cells to extract features and visualize the frequency content and periodic details of the drones' movements. The results showed that the applied transformations effectively captured the drone's Doppler components and rotational rates. The CVD, in particular, revealed valuable information about the frequency content within each Doppler cell. To automatically identify and allocate the principle features, a pre-trained CNN known as AlexNet was utilized. To observe the differences between payload cases, the activations of the cepstrogram and CVD at several layers of the deep network were visualized. The results from the different transformations were fused to support the final classification decision, enhancing the robustness of the process. The paper concludes that the applied time-frequency domain transformations, along with the CNN-based classification, offer a promising approach for detecting, tracking, and classifying micro-drones based on their payload status.

Using the UCL dataset, the authors in [37] proposed a method that utilizes the spectral kurtosis technique to extract non-stationary and non-Gaussian components from the radar echoes reflected by the drones. In this study, the spectrogram of the acquired signal was computed using different smoothing windows, and the spectral kurtosis was calculated for each frequency to emphasize non-stationarities in the frequency domain. A feature extraction algorithm was then applied, followed by normalization and dimensionality reduction using principal component analysis (PCA). The resulting feature vector was used as input to a k-nearest neighbor (k-NN) classifier for classification. Experimental results on real radar data demonstrated the effectiveness of the proposed method, achieving an average accuracy of 92.61% in identifying drone payloads.

In [1], the initial findings aimed at exploring the utilization of radar data to detect the presence of payloads carried by small drones are presented (in the UCL dataset, the DJI Phantom Vision 2+ was used that cannot be categorized in the small-size UAV class). This study focused on a challenging payload scenario. The payload is small in weight, weighing only 92g overall, and compact in size, consisting of four AA batteries attached to each of the drone's four rotor blade arms rather than appearing as a single mass below the drone. They considered flight rather than hovering scenarios, which made them capable of using

features that are related to the polynomial fitting of the drone's velocity pattern. The radar system used for the experiments was the Ancortek SDR-KIT 580B, operating at 5.8 GHz. The drone chosen was the Parrot Bebop 2, weighing 500g and sized 38.1x32.8x8.9 cm [1]. For the loaded case, four AA batteries were attached individually below the drone's arms (Figure 2.1).

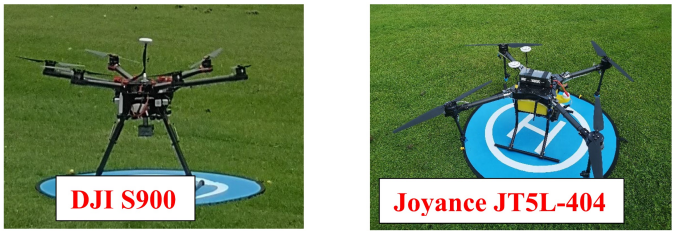


(a) Unloaded drone (b) Loaded drone

Figure 2.1: The quadcopter drone used in [1] in both unloaded and loaded cases (Copyright ©2019, IEEE)

The experiments took place at the University of Glasgow campus, with the drone flying in a narrow oval trajectory at distances of 4 to 12 meters from the radar. In order to address the binary loaded vs. unloaded classification problem, they deployed radar signatures using 12 features. The features included descriptors such as mean, standard deviation, and entropy of the spectrogram, representing pixel intensity related to the target's Radar Cross Section (RCS). Additionally, features like mean, skewness, and kurtosis of the centroid and bandwidth of the spectrogram that provide information about the drone's Doppler (velocity) over time, were used. They also introduced three coefficients of a cubic polynomial (fitted to represent changes in velocity and capture the acceleration of the drone with and without payload). This approach aims to encompass both energy spread and kinematic information, considering that payload presence can affect flight dynamics and acceleration capabilities. The classification performance was analyzed considering 4095 combinations of the 12 available features across different dwell times. Three classifiers (Support Vector Machine (SVM) with Radial Basis Function and Quadratic kernel, and Diagonal-Quadratic Bayesian classifier) were evaluated using 5-fold cross-validation. The median accuracy increased with the number of features for dwell times up to 2.5s, reaching 70-75%. The maximum classification accuracy varied from 82.5% for 0.25s dwell time to 90% for 5s dwell time, with the SVM Quadratic classifier performing the best. The confusion matrix showed that the majority of misclassifications were false positives, where unloaded drones were labeled as carrying a payload.

The authors in [2] explored the radar signatures of drones equipped with heavy payloads and dynamic payloads that generate inertial forces. The study focused on both the static and dynamic aspects of the payloads. The study involved equipping a DJI S900 hexacopter and a Joyance JT5L-404 crop spray quadcopter (Figure 2.2). The hexacopter was equipped with an electromagnet to attach steel payloads of different weights (1, 1.6, 2, and 2.5 kg), while the quadcopter carried a 5-liter water tank. Data was collected using a 24 GHz radar in FMCW mode (to collect range and Doppler data) and a 94 GHz radar in CW mode (to collect non-aliased Doppler data with higher sensitivity), simultaneously. The drones were operated at distances ranging from 20 to 60 meters away from the radars while maintaining altitudes between 5 and 8 meters. The data was processed and analyzed to extract relevant features for classification. Various radar signatures, including range-time profiles, Doppler spectra, and micro-Doppler signatures, were examined to understand the effects of different payloads on these signatures. Linear and Quadratic SVM, Linear Discriminant, and GoogLeNet were the classifiers employed in this study. The results showed that heavy payloads have a significant impact on the radar signature of drones. The range-time profiles exhibited distinct patterns for drones with heavy payloads compared to unloaded drones. Similarly, the Doppler spectra demonstrated noticeable differences, indicating the presence of a heavy payload.



(a) (b)

Figure 2.2: Two types of drones used for experimental data collectin in [2]

2.2.2 Acoustic sensor

Among all references on drone detection and classification problem, only a few have utilized acoustic data for loaded drone classification [3, 39, 40]. The authors in [3] focused on the use of DL algorithms to detect and classify UAVs based on the sound they emit when carrying payloads. To train the DL algorithms, a sufficient amount of audio data was collected, including UAV sound recordings and data augmentation techniques to increase the sample size. The dataset consisted of two UAV models: DJI Phantom 4 and EVO 2 Pro.

Recordings were done using the laptop microphone. The laptop and UAV were restricted to a maximum distance of 15m apart, and the UAV could reach a maximum altitude of 10m. A total of 2,544 samples were collected, with 1,306 samples from DJI Phantom 4 and 1,238 samples from EVO 2 Pro. The dataset was categorized into unloaded, one and two payloads (one weighing 63g and the other weighing 68g), and noise classes (see Figure 2.3 for the DJI Phantom 4 case). Data augmentation techniques, such as time-stretching, pitch scaling, time masking, and frequency masking, were applied to increase the dataset size. These techniques generated 10,140 additional data samples, resulting in a total of 12,675 samples. The extracted features included MFCC, Mel spectrogram, Chroma, Tonnetz, and contrast were used as input for the models. The CNN, Recurrent Neural Network (RNN), and Convolutional RNN models were trained and evaluated using the dataset. The results show that MFCC achieved the best performance with accuracies of 0.9493 for CNN, 0.8133 for RNN, and 0.9174 for Convolutional RNN. Overall, the CNN model achieved the best performance in audio classification, followed by Convolutional RNN and RNN. CNN effectively extracted local information from audio representations, while RNN processed sequential data. Convolutional RNN combined features from both CNN and RNN, capturing both local information and longer temporal context. The study demonstrates the potential of DL algorithms in UAV payload detection based on audio analysis.

Figure 2.3: Unloaded and loaded DJI Phantom 4 (with one and two payloads) used from experimental data collection in [3] (Copyright ©2022, IEEE)

The authors in [39] explored the use of Machine Learning (ML) algorithms to identify drones carrying payloads based on their emitted sound signals. The study proposed a feature-based classification approach and evaluated the performance of four ML models: SVM, Gaussian Naive Bayes, K-Nearest Neighbor (KNN), and a Neural Network. Audio recordings were gathered using a DJI Phantom 4 and an EVO 2 Pro drones, both with and without payloads. A total of 1232 samples were collected, amounting to a duration of 204.5 minutes. The payload used was a 500ml bottle of water. The dataset consisted of sound samples collected from loaded and unloaded drones, and various feature extraction methods were employed, including MFCC, Chroma, Mel, contrast, and tonnetz. The

results indicated that the combination of features outperforms individual features, achieving higher accuracy scores with an average accuracy of approximately 99%. MFCC and chroma were also identified as the best individual features.

Finally, in [40], remotely detecting the weight of a payload carried by a commercial drone by analyzing its acoustic fingerprint is discussed. The study investigated the relationship between the acoustic noise generated by a drone and the payload weight. Using a comprehensive experimental study involving a 3DR Solo drone, the researchers examined the variations in motor and blade speed, which influenced the acoustic fingerprint. The experimental setup involved a 3DR Solo drone, a directional microphone, a laptop for sound analysis, and the Audacity and Matlab software tools. The microphone was positioned 7 meters away from the drone, which hovered for a duration of 170 seconds. Various payload weights were tested, ranging from 0 g to 500 g in increments of 50 g. Recordings captured only the sound of the drone hovering, starting when it began hovering and ending before it landed. The analysis was focused on sound components, particularly the pitch contributed by the motor rotation speed and blades of the drone. The findings demonstrated that by using the MFCC components of the audio signal and SVM classifier, it is possible to achieve a minimum classification accuracy of 98% in detecting the specific payload class carried by the drone.

2.2.3 Vision camera

The rapid progress in the field of computer vision and advances in DL algorithms make optical sensing one of the most attractive methods for drone detection and classification. By using cameras with different Field of View (FOV), with wide field coverage and narrow FOV for higher resolution and improved identification performance, various visual (image or video) data analysis methods have been proposed for drone detection/classification problems. Among all the references which have used the vision camera for drone classification, there are only a few that considered the payload classification problem. Seidaliyeva et al. [4] proposed a DL approach to the two-class drone classification problem (loaded vs. unloaded). In this reference, the authors collected their data using a DJI Phantom 2 and captured some videos from it in both loaded and unloaded cases. In addition, because of the shortage in the number of recorded frames, some photos from drone delivery services such as Amazon and DHL were taken from some public open sources. The final dataset contained 1000 images for each class. Unfortunately, there is no specific information about the type of payloads used in the test and the dataset is not publicly available. A sample image from each class can be seen in Figure 2.4.

Figure 2.4: Sample image of loaded and unloaded drone in the dataset of [4, 5] (Copyright ©2020, IEEE)

This paper focused on computer vision-based UAV detection using the YOLOv2 algorithm. Unlike two-stage classification-based algorithms, YOLOv2 is a single-stage regression-based algorithm that simultaneously predicts classes and bounding boxes for the entire image. YOLOv2 employs techniques such as batch normalization and anchor boxes to improve accuracy and training speed. The proposed network model consisted of 23 convolution layers, including max pooling layers and skip connections. The YOLOv2-based detection approach achieved a mean Average Precision (mAP) of 74.97% and a detection speed of 46 Frames Per Second (FPS). The results demonstrated the proposed algorithm can detect loaded UAVs, although the accuracy is not high. The same authors utilized a deep residual convolutional neural network (ResNet-34) for the binary classification problem between loaded and unloaded drones in [5]. The network takes Red-Green-Blue (RGB) images of loaded or unloaded UAVs as input, and after passing through the ResNet-34 layer with pre-trained weights, it goes through a fully connected layer with 1024 neurons. The Softmax classifier completes the framework. Rectified Linear Unit (ReLU) activation function is used in the convolution layers of the framework to introduce non-linearity. The model was trained for 25 epochs to test its effectiveness. The same dataset as [4] was used for performance analysis in this paper. The model was trained using the collected dataset, and metrics such as loss and accuracy were measured for training and validation data. The results showed a training accuracy of 96% and a validation accuracy of 92.19%. The model achieved a training loss of 0.1212% and a validation loss of 0.2497%.

2.2.4 Summary of payload detection methods

The literature review on different methods employed for drone payload detection can be summarized in Table 2.1. Based on the research conducted in this field, vision cameras offer the advantage of being cost-effective and are generally applicable, even for unknown

drone models. While no specific details about the effective working distance of cameras have been reported in this field, one can infer that as long as a drone can be detected in the captured frames, it is likely that its payload can also be identified. As demonstrated in the following chapter, detecting a drone's payload with a high degree of confidence is feasible at distances exceeding 100 meters.

Table 2.1: Comparison of various existing methods for drone payload detection

	Radar	**Acoustic sensor**	**Vision camera**
Pros	Work at various light conditions, Medium/long ranges	Not high cost	Low cost, Mostly works even for new unknown drones
Cons	High cost, Required to know the drone model aprior	Too short working distance, Necessity of having not busy environment to record sound, Required to know the drone model aprior	Limitation in working light condition
Specifications	Experiments at maximum distance of around 78 meters	Experiments at less than 15 meters	No report about the working distance

2.3 Drone Localization

Localization refers to the process of determining the precise position of a drone in a given environment. Accurate drone localization is essential for enabling autonomous navigation, precise control, and coordination of multiple drones in complex scenarios. Vision-based localization techniques leverage visual data, such as images or videos captured by onboard cameras or external sensors, to estimate the drone's position. This section provides the existing literature on using vision cameras for distance estimation. It also explores the various methodologies and algorithms for drone localization using vision data.

The most challenging part of localization using 2D images captured by commercial RGB cameras is distance estimation. One fundamental challenge arises from the inherent loss

of depth information in a two-dimensional representation. Unlike three-dimensional space, where objects have varying distances from the observer, a 2D image lacks the ability to capture the true depth of the scene. This limitation makes it difficult to accurately estimate the distance to objects. Additionally, factors such as perspective distortion and varying lighting conditions further complicate the task, as they introduce further ambiguities and inconsistencies in the image data. Thus, the accurate estimation of object distance solely from a 2D image is a complex problem that demands innovative approaches and robust algorithms within the field of AI. Research on distance estimation using 2D images can be classified into two categories: methods suitable for general objects or objects other than drones, such as cars on the road, and methods specifically designed for drone distance estimation.

2.3.1 Distance estimation for non-drone objects

In [41], the authors addressed the scale ambiguity problem in classical monocular visual simultaneous localization and mapping (vSLAM) or visual odometry (VO) methods. The proposed approach suggested estimating the scale by determining the traveled distance between a set of consecutive frames rather than relying on depth maps or external information. The authors proposed an "end-to-end many-to-one traveled distance estimator" called DistanceNet, which uses a deep recurrent convolutional neural network (RCNN). The CNN part of the model learns geometric features, which are then used by the RNN to learn temporal information and estimate the traveled distance between the images. The model utilizes ordinal regression to predict the distance. Experimental results on the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago (KITTI) dataset [42] demonstrate that the proposed approach outperforms the state-of-the-art deep learning pose estimators and classical mono vSLAM/VO methods in terms of distance prediction.

The authors in [43] addressed the problem of object-specific distance estimation for autonomous driving. The authors proposed an end-to-end learning-based model that directly predicts distances for objects in images. They introduced a base model that extracts features from RGB images and utilizes Region of Interest (ROI) pooling to generate a fixed-size feature vector for each object. The feature extractor processes RGB images using popular network structures, such as VGG16 or ResNet-50, to extract a feature map. These features are then passed through a distance regressor to predict the distance. The models are trained using a combination of classification and distance regression losses. An enhanced model is also proposed, which includes a keypoint regressor and a projection loss

to improve distance estimation, especially for objects close to the camera. Their experimental results showed that the proposed methods outperform alternative approaches on object-specific distance estimation, particularly for challenging cases such as objects on curved roads.

In [44] a 3D spatial localization method is proposed using a single camera with temporal-difference image processing. The method utilized a ring of light-emitting diodes (LEDs) embedded on the target and estimated the target location based on the diameter and central location of the ring's image on the image sensor. The proposed method does not require knowledge of camera hardware parameters. However, the necessity of utilizing the LEDs on the object (which is not necessarily available all the time) is the main drawback of their method. The paper introduced a proposed network structure called the Volterra series method (VSM). The experimental results demonstrated the effectiveness of the proposed method for low-power outdoor unmanned aerial vehicle localization and indoor robotic navigation.

The authors of [45] presented a ML setup for estimating the distance between a monocular camera and objects in the camera's field of view. The proposed system, called DisNet, is based on a Multi Hidden-Layer Neural Network. The network is trained using supervised learning, where manually calculated parameters of object bounding boxes obtained from the YOLO object detector (including BB width, height, and diagonal in the number of pixels) are used as input features and accurate 3D laser scanner measurements of object distances are used as outputs. The evaluation of the DisNet system on railway and road scenes demonstrates its general applicability to different types of monocular cameras. The main drawback of their method is that it only relies on some BB features such as width which can introduce inaccuracies in cases such as occlusion or when the object is oriented in the captured image (and thus, the size of horizontal BB is not as the same as the not oriented case).

A two-stage DL-based framework was proposed in [46] for absolute distance prediction based on object detection and monocular depth estimation using a single image. The framework consists of two deep networks: YOLO-v5 for object detection and a deep autoencoder network for depth estimation. The YOLO-v5 network was trained using supervised learning, while the depth estimation network was trained using self-supervised learning. The autoencoder architecture consists of encoders and decoders. For both the depth and pose networks, the encoders utilized the ResNet pre-trained weights to extract features and represent the input images. Specifically, ResNet-50 served as the backbone network for depth prediction, while ResNet-18 was employed for pose estimation. Additionally, the proposed approach incorporated Graph Convolutional Networks (GCN). By using the power of GCN, the model could learn features by examining neighboring nodes

and enhance the precision of depth estimation. The proposed framework was evaluated on real outdoor images and achieved promising results with an RMSE of 0.203 for absolute distance estimation.

The authors in [47] presented an approach to integrate sparse radar data into a monocular depth estimation model for autonomous vehicles. The authors proposed a preprocessing method to address the sparseness and limited field of view of radar data, resulting in increased data points with reduced error. They also introduced a DL-based method, inspired by the deep ordinal regression network, to estimate dense depth maps from monocular RGB images and sparse radar measurements. The radar data was incorporated into the network using a late fusion approach, where the sparse 2D points were converted to height-extended 3D measurements. The proposed method achieved state-of-the-art performance in both day and night scenes on the nuScenes dataset.

A multi-task learning network called SynDisNet was presented in [48] to improve self-supervised monocular distance estimation for fisheye and pinhole camera images. The proposed method introduced a distance estimation network architecture that combines a self-attention based encoder with robust semantic feature guidance to the decoder. It also integrated a generalized robust loss function, eliminating the need for hyperparameter tuning with the reprojection loss. To address artifacts caused by dynamic objects violating static world assumptions, a semantic masking strategy was employed. The proposed method achieves a 25% reduction in root mean square error (RMSE) compared to previous work on fisheye cameras and achieves state-of-the-art performance on the KITTI dataset using a pinhole model. However, one drawback is that the method relies on semantic segmentation, which may introduce errors if the segmentation is inaccurate.

In autonomous driving tasks, accurately determining the distance between vehicles and obstacles is crucial for obstacle recognition and avoidance. The authors in [49] focused on detecting 3D bounding boxes for objects. In this paper, a novel approach was proposed that combines a 2D object detection network with a depth estimation network based on YOLO-v4. By adding a single depth channel to the 2D object detection model, the proposed method achieved improved performance and faster detection speed compared to traditional 3D detection models. The contributions of this work included a simplified model architecture and a novel loss function for depth estimation. The proposed method achieved accurate depth prediction in both near and far ranges through a combination of the Huber loss and inverse depth loss functions.

The study [50] focused on designing a computer vision algorithm for estimating the absolute distance of objects using RGB cameras. The proposed algorithm utilized the YOLO-v3 object detection algorithm as its base and extended it to include distance es-

timation. The proposed method, called Dist-YOLO, enriched the training dataset with distance information, extended the prediction vectors to include distance, and updated the YOLO-v3 loss function accordingly. The study demonstrated that Dist-YOLO achieves more accurate bounding box detection compared to the baseline YOLO algorithm while using the same backbone capacity.

This literature review in [51] explored recent advancements in DL-based monocular depth estimation (MDE) for computer vision applications. The paper discussed the advantages of MDE over stereo-based depth estimation due to its low cost and simplicity. It compared MDE models based on different input data shapes and training approaches, including mono-sequence, stereo sequence, and sequence-to-sequence. The review analyzed various datasets and evaluation metrics used in DE models. Additionally, it discussed different training approaches, such as supervised learning, unsupervised learning, and semi-supervised learning. The paper reviewed popular datasets for MDE, such as KITTI, NYU Depth-V2, and Cityscapes, and evaluation metrics commonly used to assess the performance of DE models.

Finally, a self-supervised learning-based method for distance estimation was proposed in [52]. The authors introduced Lite-Fast YOLO-v5, a network structure that combines the efficient Shufflenetv2 backbone with multi-scale prediction to improve detection speed without sacrificing accuracy. They also proposed a self-supervised learning approach for distance estimation, utilizing sequence input images and a new reconstruction loss function. The proposed method outperformed previous approaches and is evaluated on the KITTI and CCP datasets.

2.3.2 Distance estimation for drones

In addition to the previously mentioned references, there are some limited works on the UAV distance estimation using vision camera. In [53], the authors focused on the development of a vision-based object detection method using deep learning-based distance estimation for mid-air collision avoidance in UAVs. The proposed approach utilized a monocular camera as the sole sensor to detect and estimate the distance of a fixed-wing intruder. A multi-stage object detection scheme was proposed, combining background subtraction and a YOLO-v3, to detect and classify objects in both short and long distances. After detection, two different DL approaches, namely CNN and Deep Neural Network (DNN) with the DisNet regression model [45], were used for distance estimation comparison. In the CNN distance estimation approach, a 5-layer network architecture based on Visual Geometry Group (VGG)-16 was used as the feature extractor with the number of filters

equal to 64 to 512 at different stages. The flatted 4608-element feature vector was then passed to the regression network which was composed of 4 layers of various fully-connected blocks. Also, the authors used the BB specifications (including its width and height) extracted from YOLO-v3 as the inputs to DisNet [45] for distance estimation. However, to accurately detect the attitude and distance of the UAV, the authors modified the last three parameters of DisNet (average height, width, and breadth of the object) to instead include the roll, pitch, and yaw angles. The performance of both distance estimation networks was compared and the CNN one was selected because of its robustness and also, less dependency on the size of BB. The Blender software was utilized as the synthetic data generator. The prepared dataset contained just one type of fixed-wing drone. Additionally, for experimental data collection, a fixed-wing Sky-surfer UAV model was used.

The authors in [6,54] presented a single-camera-based optical spatial localization system for UAVs. The system consisted of a blinking LED ring mounted on the UAV as a marker, an event-based dynamic vision sensing (DVS) camera for image sensing, a band-pass optical filter for object detection, and a 3D localization algorithm running on a base station. The proposed configuration can be seen in Figure 2.5.

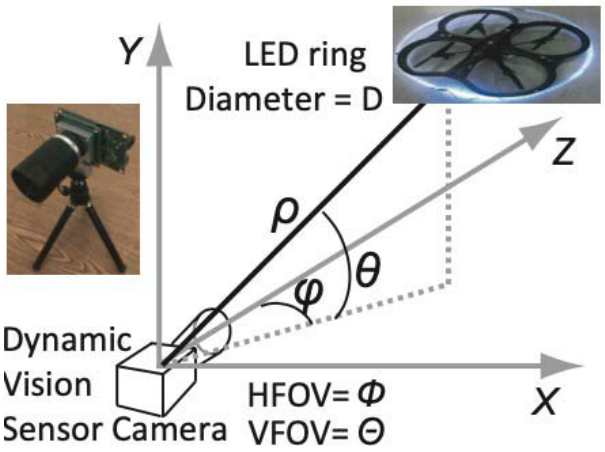


Figure 2.5: The proposed configuration in [6] (Copyright ©2022, IEEE)

The proposed system aimed to achieve high-speed and accurate localization without relying on GPS or Wi-Fi signals. The event-based DVS camera helps identify the blinking LED marker from the image stream, and the localization algorithm calculates the UAV's 3D location based on the known physical parameters of the marker. Experimental tests using a motion capture system validated the system's real-time performance. The proposed system has the advantage of low power consumption on the UAV and low computing overhead at

the base station. However, it entirely depends on using the LED on the platform, which is not feasible when we encounter a non-owned drone.

2.4 Summary

This chapter's literature review provides a comprehensive overview of the existing research, methodologies, and advancements in drone localization and payload detection.

Regarding payload classification, all of the radar-based methods utilized various time-frequency analysis tools to process the reflected signal. Additionally, the micro-Doppler signature played a crucial role as either the key feature or one of the main features in solving the binary classification problem of loaded versus unloaded drones. In terms of the maximum classification range, most radar-based methods employed the UCL dataset (except for two methods), with a maximum distance of approximately 78 meters between the radar and the UAV. However, the remaining two references reported maximum distances of around 60 meters and distances less than 12 meters, respectively. It is worth noting that, except in [2], only one drone model was used in the experimental tests.

As for the acoustic-based payload detection methods, the primary feature used for the binary classification problem was the MFCC. With regards to the drone model, all methods, except one, focused on a single drone model in their research. The exceptional method utilized EVO 2 Pro and DJI Phantom 4. Regarding the distance, although explicit reporting is lacking (except in one of the references), the maximum range is very limited, with one paper reporting a range of less than 7 meters. The reason for this is that sound recording is only feasible at shorter ranges, making it impractical for long-range scenarios.

Finally, in the vision-based payload detection, two deep networks were employed as classifiers. Unfortunately, there is limited information available regarding the dataset used in their work. However, as reported, a portion of the dataset was recorded using DJI Phantom 2. Based on the provided sample images, it seems that the drone was positioned relatively not far to the camera during the tests.

In the field of drone localization, this chapter explored the existing literature on drone localization using vision-based techniques. One significant challenge in vision-based localization is distance estimation, as two-dimensional images lack depth information. Various methodologies and algorithms have been proposed to address this challenge which were reviewed in this chapter. However, there are limited works on UAV distance estimation using vision cameras. One study proposed an approach that utilized a monocular camera as the sole sensor and employed multi-stage object detection schemes combined with deep

learning models for distance estimation. One significant limitation of this study is the use of a complex network for distance estimation, which imposes a high computational load and is not suitable for real-time implementation due to its resource-intensive nature. Another research effort focused on developing a single-camera-based optical spatial localization system for UAVs, which utilized a blinking LED marker and an event-based dynamic vision sensing camera. The main drawback of that study is that it entirely relies on some specific hardware installation such as LEDs and dynamic vision sensing cameras.

Chapter 3

Payload detection

3.1 Introduction

Detecting the payload of drones is an important aspect of drone security, as it can help prevent potential threats and illegal activities. The payload refers to any item or device that is carried by the drone, such as cameras, weapons, or explosives.

Recent advances in deep learning have led to the development of more advanced techniques for object classification using visual features. By visual features, we mean the various characteristics of images or videos, such as color, texture, and shape, that can be used to identify objects within them. Changes in the appearance of drones due to the existence of payloads can be used as a feature by deep learning methods for recognizing loaded drones. This approach involves training a deep neural network to recognize specific features or patterns associated with different types of payloads. However, the main drawback is that deep networks require a massive amount of data for the training phase. Collecting such an amount of data using practical tests is generally a costly and time-consuming task. Thus, in these situations, the simulation would be a better choice.

In this chapter, three variations of the ResNet and EfficientNet architectures are employed as classifiers to differentiate between loaded and unloaded drones. Specifically, the ResNet-34, 50, and 101 models and EfficientNet-B2 are utilized, with a particular focus on the ResNet-34 model due to its shallower architecture. The ResNet-34 model requires fewer computational resources, making it well-suited for real-time implementation and applications where computational efficiency is a priority. Additionally, we created the first simulated air-to-air vision dataset for the loaded vs. unloaded drone classification problem. Five types of drones with two forms of payload (attached and hanging,

and with different colors) have been simulated in this dataset. The dataset is recorded in three different environments (with various weather conditions, including sunny, rainy, and snowy cases) to provide sufficient diversity in the background of the drone in the recorded videos. All the captured frames are annotated, and the XYZ coordinates of the camera and the drone in each frame are provided. Finally, the trained ResNet and EfficientNet networks using the simulated data are also applied to the recorded practical data. The promising results on the practical data demonstrate the superiority of our dataset. The simulated dataset as well as the codes for classification are publicly available at www.github.com/CARG-uOttawa/loaded-unloaded-drone-dataset.

3.2 Problem definition

Drone payload detection is an important research area that aims to address the classification of drones into two distinct categories: loaded and unloaded. The ability to accurately identify whether a drone is carrying a payload or not is of great significance in various applications, including security, surveillance, law enforcement, and border control. This binary classification problem forms the foundation for differentiating between drones with potential security risks, such as those carrying hazardous materials or unauthorized cargo, and drones that are operating within legal boundaries. As it was mentioned in the previous chapters, drone payload detection algorithms utilize a range of sensor inputs, including visual imagery, radar, and acoustic sensors. These sensors capture valuable information about the drone's appearance which can be leveraged to differentiate between loaded and unloaded drones.

Let $\mathbf{X}$ be the input data comprising sensor measurements, such as visual imagery captured by the drone payload detection system. Given the input data $\mathbf{X}$, the objective is to classify the drone into one of two classes: loaded (class 1) or unloaded (class 0). The raw input data, denoted as $\mathbf{X}$, can undergo various feature extraction techniques to capture relevant information for drone payload detection. These extracted features encompass visual characteristics, shape descriptors, motion patterns, or any other discriminative features. The purpose of feature extraction is to transform the raw data into a representation that is more suitable for classification tasks. In the context of classical machine learning, these extracted features are then used to train a classifier that can distinguish between the two classes: loaded and unloaded drones. However, with the emergence of deep learning, a new paradigm has been introduced. DL-based classifiers leverage deep neural networks, which are capable of automatically learning and extracting high-level features directly from the raw data. Unlike traditional ML approaches that rely on handcrafted feature engineer-

ing, DL classifiers have the ability to learn hierarchical representations of the input data. This allows them to capture intricate patterns and relationships that may not be easily discernible through manual feature engineering. By employing deep neural networks for payload detection, we can potentially achieve higher accuracy and robustness compared to traditional ML methods. The network learns to automatically extract and utilize the most relevant features for discriminating between loaded and unloaded drones, thereby improving the classification performance.

3.3 ResNet and EfficientNet architectures

As it was mentioned, three variants of ResNet and EfficientNet architecture are utilized as the classifier in this study, with a particular focus on the ResNet-34 model. The ResNet architecture has emerged as a groundbreaking deep learning model, revolutionizing the field of computer vision and image recognition. Developed by He et al. in 2016 [7], ResNet stands for Residual Network, and it addresses a fundamental challenge faced by deep neural networks - the degradation of accuracy with increasing depth. ResNet-34 is one variant of the ResNet family, specifically composed of 34 layers, and it has demonstrated remarkable performance in various computer vision tasks.

In traditional deep neural networks, the network depth was considered a critical factor in achieving better performance. However, as the network depth increased, the problem of vanishing gradients and the degradation of accuracy became more prominent. Deeper networks were unable to achieve improved accuracy and instead suffered from diminishing returns or even degradation in performance. This limitation hindered the development of deeper and more powerful neural network architectures. To overcome this challenge, ResNet introduced the concept of residual learning. The key innovation of ResNet is the introduction of skip connections, also known as shortcut connections or identity mappings, that allow the network to learn residual functions. These skip connections enable the network to directly learn the difference between the input and output of a specific layer, effectively bypassing the need to learn the entire transformation. This mechanism facilitates the training of much deeper networks without sacrificing accuracy, and it enables the development of architectures with hundreds or even thousands of layers.

The basic structure of ResNet-34 is inspired by the famous VGGNET [55]. The architecture of ResNet-34 is characterized by its deep structure, which consists of 34 layers. These layers are organized into several building blocks, each containing a set of convolutional, batch normalization, and activation functions. The distinguishing feature of ResNet-34 is

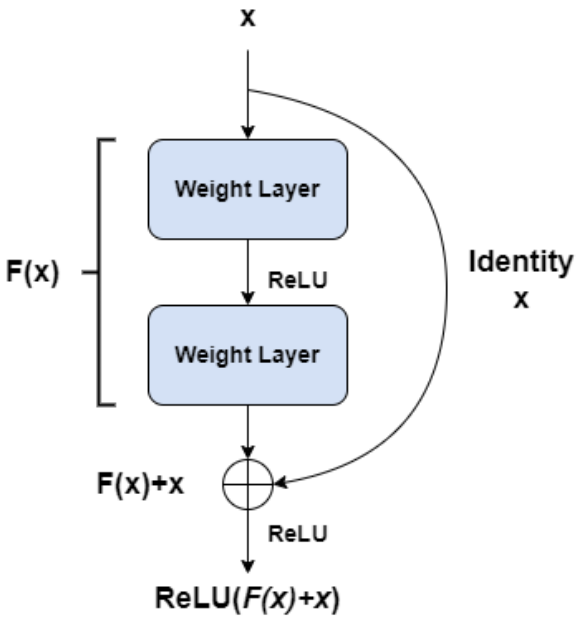


Figure 3.1: A building block of a residual network

the introduction of residual connections, also known as skip connections or shortcut connections. The basic building block of ResNet-34 is the residual unit, which consists of two convolutional layers with a batch normalization and ReLU activation function in between. These residual units are stacked together to form the overall architecture. The structure of the residual block can be seen in Figure 3.1.

The whole architecture of the ResNet-34 is also shown in Figure 3.2.

In this study, the application of the ResNet-34 network is explored to address the binary classification problem of distinguishing between loaded and unloaded drones. Obviously, deciding whether a drone is loaded or not is the next step after drone detection. Thus, it

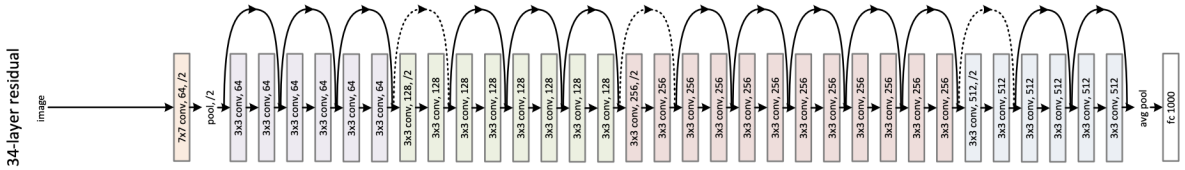


Figure 3.2: The architecture of ResNet-34 network (adapted from [7])

is assumed that the bounding box information of the detected drones is available from an object detector network. Here, just as the same as the previous chapter, YOLO-v7 is used as the detector (with the same trained weights that were explained in Chapter 1). To prepare the input images for the ResNet-34 network, the entire image frame is cropped around the detected bounding box. The resulting cropped image is then resized to a fixed size of 255×255 pixels. For the resizing process, the default interpolation technique provided by the *torchvision* library is utilized, which employs the bilinear method. Consequently, the resized image representing the drone's region of interest is fed into the ResNet-34 network for payload detection. It is worth mentioning that the imported images are set to the *JPEG* format, which is one of the most well-known compressed formats.

Although the results section will demonstrate the successful performance of the ResNet-34 architecture in accurately classifying loaded and unloaded drones, the large size of the simulated dataset developed for this thesis provides the opportunity to explore deeper variants of the ResNet model. The ResNet-50 and ResNet-101 architectures, also belonging to the ResNet family, have proven to be highly effective in various computer vision tasks, including image classification. With 50 and 101 layers, respectively, these architectures can capture intricate features and learn discriminative representations, making them well-suited for complex classification problems (such as the works on drone-related topics [56–60]). In both ResNet-50 and ResNet-101 architectures, the initial 7×7 convolutional layer and the subsequent pooling layer are identical to those in ResNet-34. However, several changes are made in the subsequent layers. In ResNet-50, there are 3 layers with 1×1 kernels and 256 output channels, followed by 4 layers with 1×1 kernels and 512 output channels, 6 layers with 1×1 kernels and 1024 output channels, and finally, 3 layers with 1×1 kernels and 2048 output channels. These layers are inserted into the four levels of convolutional filter sets, as shown in different colors in the ResNet-34 architecture diagram (Figure 3.2). The same modifications are applied to ResNet-34 to create ResNet-101, except for the third set, which includes 23 repetitions of the layers instead of 6 in ResNet-50. These additional layers increase the complexity of the network during training, requiring the learning of more weights. However, this increased complexity enables the extraction of more nonlinear features from the data.

To enhance the performance of CNN networks across various tasks, a common strategy involves employing a scale-up approach. Scaling can occur in several dimensions, including the network's depth, width, and resolution. In their work [61], the authors introduced a balanced approach to simultaneously scale up all three of these dimensions. This innovative methodology gave rise to a class of deep networks known as EfficientNet. These network models are compared to other traditional architectures, including ResNet. According to the results reported in [61] on the well-established *ImageNet* dataset, the EfficientNet-

B2 architecture demonstrates superior performance compared to the ResNet-34 classifier while maintaining an equivalent or even less computational load. Based on this observation, Efficient-B2 has been chosen as one of the deep classifiers in this study in addressing the drone payload detection problem.

The architecture of EfficientNet-B2 consists of multiple blocks, where each block is a stack of layers. These blocks are organized in a hierarchical fashion, with each subsequent block processing features at a higher level of abstraction. This hierarchical structure allows the network to capture increasingly complex patterns in the input data. EfficientNet-B2 also features mobile inverted bottleneck blocks, which help to reduce computational complexity while preserving model capacity. These blocks consist of depth-wise convolutions, point-wise convolutions, and expansion layers. In addition to these components, EfficientNet-B2 incorporates various architectural innovations, such as squeeze-and-excitation blocks, which enhance feature recalibration, and a progressively increasing resolution strategy. EfficientNet models, including B2, have demonstrated impressive results on various benchmark datasets and have been widely adopted in the field of computer vision due to their efficiency and effectiveness.

In the subsequent sections, detailed information about the simulated dataset prepared specifically for drone payload detection will be presented. Furthermore, the classification results obtained using the ResNet and EfficientNet networks will be thoroughly examined and analyzed. It is important to note that all the forthcoming results will be presented on a frame-by-frame basis. Whenever a detection indicates the presence of a drone in the scene, the corresponding BB area of the frame will be fed into the classifier, which determines whether the drone is loaded or not. In this strategy, even in the case of false detections (i.e., incorrectly identifying other objects such as birds as drones), the BB area containing the object is processed by the network, resulting in a decision of either "loaded drone" or "unloaded drone," which may be incorrect. As a topic for future work, a method should be devised to address these situations. For instance, the payload detection network can be activated only for the detected targets with high confidence from the detector network. Alternatively, to handle false alarm occurrences in the detector, the payload classifier could be triggered after several detections from the detector network (since false alarms tend to sporadically appear in different parts of the recorded frame). It is also possible to monitor the classifier's output over time to determine its validity (assuming that, for objects other than drones, the classifier's output fluctuates between loaded and unloaded decisions due to the object's shape).

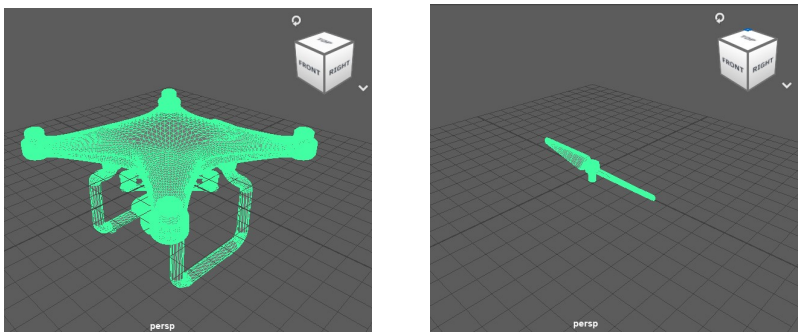
3.4 Simulated unloaded-loaded drone dataset

As it was explained in the introduction, there is not any available public vision dataset for the loaded vs. unloaded drone classification problem. AirSim© provides this opportunity to simulate flying drones in different environments. There is only one simple quadcopter available as the default model that can be utilized in the simulations. However, as an open-source tool, there is the possibility to import other drone models into AirSim© (although it is not straightforward work). In this study, various models of drones in both loaded and unloaded forms have been imported to this environment in order to accomplish some flying scenarios and collect the required dataset. In this section, the procedure of creating the simulated air-to-air dataset including the process of importing new drone models into AirSim© will be explained. Besides, some statistics about the simulated dataset and an explanation of uploaded files on the shared link will be presented.

3.4.1 Procedure of importing custom drone model into AirSim

In order to import the desired UAV model into AirSim©, first of all, some preparation should be done on the 3D model of the drone. The drone is modeled with two distinct parts in AirSim© which are the propellers (each of them, individually) and the body. Generally, the available 3D models for the drones that are accessible through websites such as *cgtrader*, include several components which totally form the whole model. Thus, in order to import the 3D model into AirSim©, they should be combined as a whole object called *body* (without the propellers). Additionally, only one of the propellers should also be extracted for constructing the AirSim© drone model. Various 3D design tools and software can be utilized to extract these two objects from the original 3D model. In this study, we use Autodesk Maya for this purpose [62]. At first, the propellers and the remained parts of the drone's 3D model should be separated and saved in two different files including the body and propeller (only one propeller should be saved in the file). Then, by using the "*combine*" option, the parts should be merged to form a single object as the body. Finally, these 3D models should be exported as *fbx* files into AirSim©. As an example, Figure 3.3 shows the two 3D shapes that are prepared for DJI Phantom.

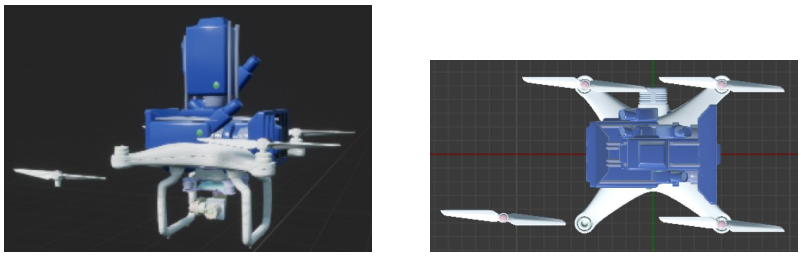
After importing the *fbx* files of the prepared 3D objects into AirSim©, the drone model should be constructed based on them. For this purpose, the easiest way is to duplicate the default drone model in AirSim© and then, replace the *body* and *propeller* parts with the imported objects. In the editing window for the *blueprint* of the drone, the body and propeller parts can be replaced by the prepared objects. It should be noted that in this



(a) Body object (b) Propeller object

Figure 3.3: Prepared 3D objects to be imported into AirSim©

step, some changes, including rotation, translation and scaling, may be needed to adapt the 3D objects to AirSim©. In addition, the color and material of the objects can be selected in this step. Figure 3.4 shows adding the last propeller to the new drone model in AirSim© (the blue 3D parts are just some AirSim© internal shapes to show the sensors mounted on the drone). Finally, the imported drone can be saved as a new *blueprint* model in AirSim© and be called by putting its name in the *settings* file during running AirSim©.

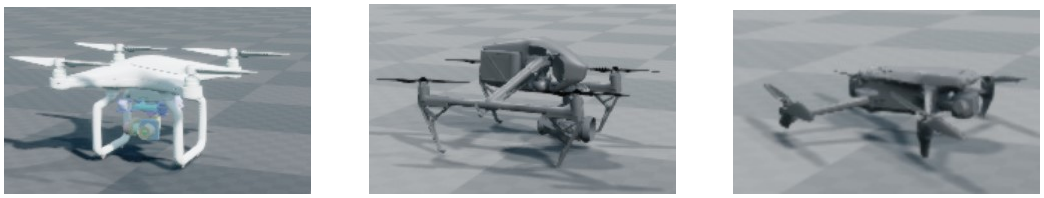


(a) Prespective view (b) Top view

Figure 3.4: Adding the last propeller to the new drone model in AirSim©

Based on the explained procedure, 5 types of drones have been imported into AirSim© in our dataset (Figure 3.5) including 4 DJI models [63] and also, a generic drone model. From this point on, this model is called Quadrotor in the thesis, and its width with widely spread propellers is assumed to be around 70 cm (a bit bigger than DJI Phantom). These drones are selected based on their size category. In other words, DJI Phantom and Quadrotor can be considered as two models in the large-size category. DJI Inspire is a sample of the mid-size drones and, DJI Mavic and DJI FPV can be categorized in the small-size

group.



(a) DJI Phantom (b) DJI Inspire (c) DJI Mavic

(d) DJI FPV (e) Quadrotor

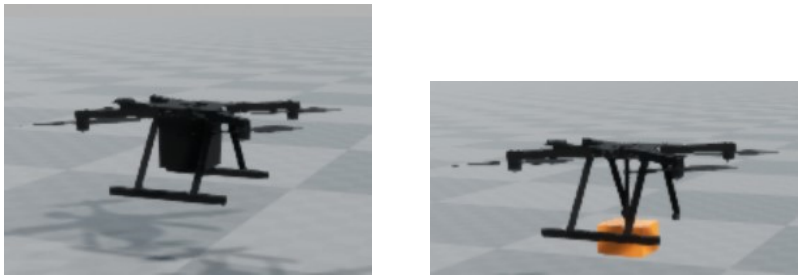
Figure 3.5: Different drone models simulated in the dataset

3.4.2 Payload simulation

As mentioned in the previous sections, this chapter presents a simulated dataset for the classification problem of loaded vs. unloaded drones. To ensure diversity in the dataset, various types of drones were simulated, along with different forms of added payloads. Specifically, box-shaped payloads were attached and hung from the simulated drone models.

For models equipped with external cameras such as DJI Phantom, Inspire, and Quadrotor, the payload replaces the camera's position. The payload is simulated in two colors: black, which is similar to the drone's body color, and orange. These colors were chosen based on the level of distinction between the payload and the main body. The orange color provides a higher level of contrast, resulting in an easier classification problem, whereas the black color is more similar to the body color, making it more challenging to differentiate from the overall body.

Importing loaded drones follows the same procedure as regular drones, with the only difference being that the payload needs to be initially created using the *Curves/Surfaces* option in Autodesk Maya. It is then combined with the body object to form a complete 3D model of the loaded drone. Two examples of simulated loaded drones with different forms and colors are depicted in Figure 3.6.



(a) Attached box payload (b) Hanging payload

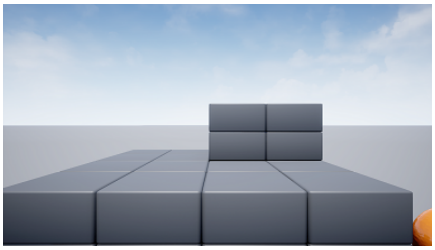
Figure 3.6: Two forms of loaded drones in the simulated dataset

3.4.3 Simulated Environments

To utilize imported drones in AirSim©, it is necessary to create custom environments as the pre-built binary files on the AirSim© website are incompatible. For our dataset, we select 3 environments including simple *Blocks* environment, as well as *Landscape mountains* and *City Park Environment-Lite*. The blocks environment can be downloaded with AirSim© itself, while the other ones are two free environments in Unreal marketplace [64]. The *Blocks* environment can be considered as the simplest case in which the drone has a flight path with a sky background almost all the time (except those times that it is near the blocks). *Landscape mountain* environment is selected as a natural case in which we have trees, rocks, and also a lake as the background. Finally, the *City Park* environment provides a mixture of natural items such as trees and some man-made objects such as a building. As it was mentioned, we here encounter a classification problem based on the vision dataset. In such a problem, the classification accuracy can be affected by the type of background. By considering this point, we tried to simulate various backgrounds in the dataset. A sample scene of all these environments can be seen in Figure 3.7.

3.4.4 Statistics about the simulated dataset

Considering the approach described above, the specification of the simulated dataset will be explained in this part. In practical situations, various types of drones may be used to carry the desired payload. Thus, in the published dataset, five types of drones including 4 DJI models and the generic Quadrotor model have been imported into AirSim© environment. Considering other simulation conditions explained in the previous parts, a brief explanation of dataset specification can be seen in Table 3.1. In each case, a video of around 1000 frames with a resolution of 1080×1920 pixels and a FOV of 82 degrees has



(a) Blocks

(b) Landscape mountains

(c) City park - Lite

Figure 3.7: Different environments used in the simulated dataset

been recorded, and the annotation file (using the "object detection" feature of AirSim©) has been provided. Figure 3.8 shows an example of the shape of Quadrotor in both loaded and unloaded cases (based on the information of the bounding box around the drone in various frames, the images are cropped, and then, all of them are resized to have a square formation). At far distances, the size of the detected drone in the image frame becomes quite small, often occupying an area with a width of less than 20 pixels. However, all the images are resized to a fixed size of 255 by 255 before being input to the classifier network. This resizing process leads to the appearance of blurred images, as illustrated in Figure 3.8.

Table 3.1: Dataset explanation

Types of drones	DJI Phantom, DJI Inspire, DJI Mavic, DJI FPV, Quadrotor
Environment	Simple block Env., Mountain landscape, City park
Weather condition	Sunny, Rainy, Snowy
Payload condition	Unloaded, Attached load (box), Hanging load

As mentioned in the previous parts, air-to-air video recording of the observed drone helps to have various view angles of it (compared to the case when the camera is on the ground and there is only a bottom view angle of the drone). To illustrate this advantage, Figure 3.9 shows the captured image of two types of drones for various height differences

Figure 3.8: An example of the shape of Quadrotor in the collected dataset for the loaded and unloaded cases

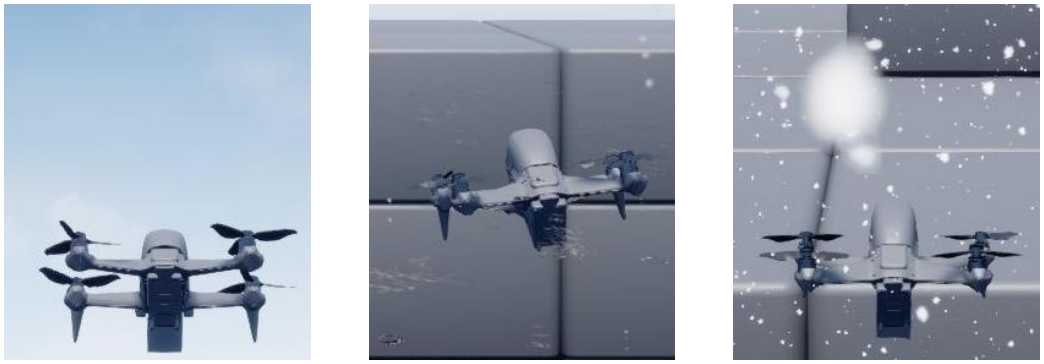
between the camera and the observed drone.

(a) Quadrotor (b) DJI Mavic

Figure 3.9: Captured image of two types of drones for various height differences between the camera and the observed drone (left: up view, middle: same height, right: bottom view)

The effect of weather on the recorded videos for sunny, rainy, and snowy weather can be seen in Figure 3.10. It can be observed that in rainy and snowy weather, it is probable that the drops fall in front of the camera and cover some parts of the recorded scene. Such occlusion can be more critical in cases where the drone is far from the camera and thus, because of its small size, is partially covered by the drops. It is worth mentioning that although various weather conditions have been simulated in the dataset, these simulations cannot fully replicate the intricacies of real-world weather conditions. For example, achieving realistically blurred images in a simulation environment, as may occur during a blizzard, can be quite challenging. Consequently, this underscores the importance of collecting real-world data for a thorough investigation of the system's final performance.

Figure 3.11 shows the distribution of the collected dataset for DJI Phantom and DJI



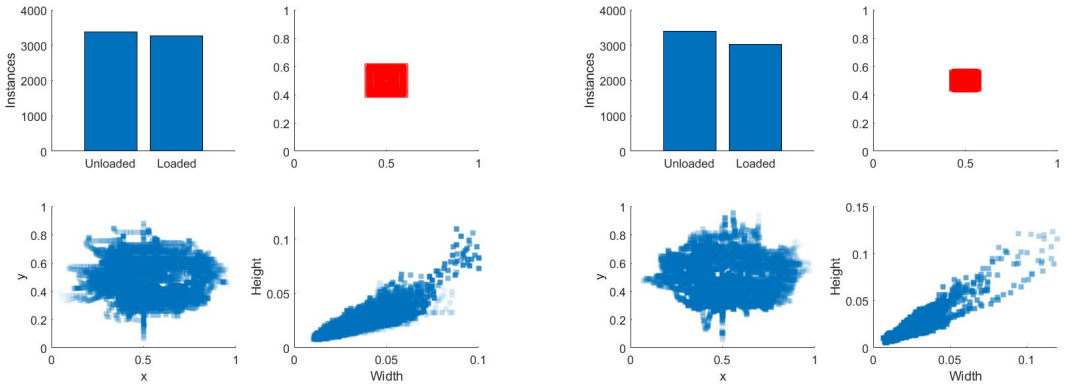
(a) Sunny (b) Rainy (c) Snowy

Figure 3.10: The effect of weather condition on the captured images

FPV as samples of two different size classes (for six scenarios and a total of around 6000 frames including loaded and unloaded drone in the sunny, rainy, snowy weather in one of the environments). In both subfigures, the top-left plot shows the histogram of the loaded and unloaded classes in the collected dataset. As can be seen, for both drones (and also, for other drones in the dataset), we have a uniform distribution for the number of samples for each class. The bottom-left figure shows the scatter plot of the location of the drone in the normalized frame region. This plot shows the drone is placed almost in all parts of this region. The top-right subfigure shows the normalized ground-truth bounding box width and height for various frames. This information shows how big is the size of the drone in the dataset. In this figure, the normalized bounding boxes with respect to the size of the image frame (1080×1920 pixels) for all the frames, are plotted. In addition, the size of both drones can be compared, and as it can be seen, on average, DJI Phantom has a bigger size than DJI FPV. Finally, the bottom-right subplot shows the scatter plot of the size of the bounding box, and how big is the size of the drone along the vertical and horizontal axis.

3.5 Results

This section presents the performance results obtained for both the simulated dataset and the experimental data.



(a) DJI Phantom (b) DJI FPV

Figure 3.11: Distribution of the collected dataset including the number of training images (top-left), illustration of ground truth bounding boxes (top-right), and scatter plots of objects' locations (bottom-left) in the frames and their sizes (bottom-right)

3.5.1 Simulated data

Due to the vast simulation conditions considered in the simulated dataset in Section 3.4.1 (including the drone type, as well as payload and environmental conditions), various problems can be defined. Based on the portion of the dataset one considers for the problem and the training and testing phases, some of the suggested problems are:

- Loaded-vs-unloaded drone binary classification just for a single type of drone (the simplest case) in just one environmental condition or in the case of various environment/weather condition
- Loaded-vs-unloaded drone classification for the general case of several drone models in the dataset
- Classification among various payload forms for the single and/or multiple types of drones (unloaded vs. $\text{payload}_{\text{attached}}$ vs. $\text{payload}_{\text{hanging}}$)

This section has considered and solved some of these problems using the deep network structure explained in the previous section. The hyperparameters of ResNet-34 have been set based on the values reported in [5]. The number of epochs and batch size are defined to be 25 and 64, respectively. Also, the value of the learning rate has been set to 0.03. Finally, the Stochastic Gradient Descent (SGD) optimizer has been used due to the better

results reported using it compared to other optimizers. Using these settings, the results of classification for some of the defined problems in the previous part are given in Table 3.2. For each problem, the data is divided into training, validation, and testing sets, with proportions of 70%, 15%, and 15%, respectively. To determine the number of frames in each row, as previously mentioned, approximately 1000 frames are simulated under each specific condition. For instance, in the second row of the table, there are a total of 6000 frames (3 weather conditions x 2 classes x 1000 frames per condition). These frames are further allocated as follows: 4200 frames for training, 900 frames for validation, and 900 frames for testing. Similar to previous works on the loaded vs. unloaded drone classification using vision data [4, 5], ResNet-34 shows highly accurate results on the binary loaded vs. unloaded drone classification problem.

Finally, it can be observed from this table that, as we expected, better classification results are achieved for the simpler cases. The classification accuracy decreases with an increase in the complexity of the background and environment (compare rows 3 and 4), weather conditions (compare rows 1 and 2, or 6 and 7), or a decrease in the size of the observed drone (compare rows 2 and 9).

Table 3.2: Classification results of the simulated dataset using ResNet-34

	Case	Acc.
1	Quadrotor, Blocks env., Sunny (Payload with orange color)	1
2	Quadrotor, Blocks env., All weather (Payload with orange color)	0.9980
3	Quadrotor, Blocks env., Sunny	1
4	Quadrotor, City park env., Sunny	0.9916
5	Quadrotor, Blocks & City park env., Sunny	0.9941
6	DJIMavic, Blocks env., Sunny	1
7	DJIMavic, Blocks env., All weather	0.9883
8	DJIMavic, Blocks & City park env., Sunny	0.9609
9	DJIFPV, Blocks env., All weather	0.9644

One of the primary challenges associated with using deep learning-based networks is their ability to generalize and perform well when applied to datasets with distinct features compared to the training data. In the current application, the synthetic data used for training predominantly featured payloads in orange, black, or gray colors. Consequently, it is important to assess how the trained network performs on a dataset featuring payloads of different colors. To investigate this, a dataset comprising 700 frames was generated,

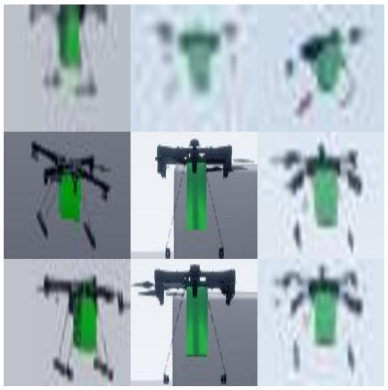


Figure 3.12: Some sample images for the case of green box as the payload

simulating a loaded Quadrotor in the Blocks environment. Unlike the previously recorded data, the payload in this dataset was assumed to be green. Sample images from this dataset are presented in Figure 3.12. Applying the network that was trained on the earlier recorded data (featuring orange, black, or gray boxes for loaded drones) to this new dataset with a green box payload yielded a classification accuracy of around 0.95. This result demonstrates the trained network's ability to successfully distinguish payloads of colors other than those seen in the training data.

3.5.2 Experimental data

While simulations provide a controlled environment for training and testing deep networks, the performance of these models on real-world data remains an essential consideration. Real-world scenarios often introduce complex factors such as varying lighting conditions, occlusions, and perspective changes, which may impact the network's ability to generalize. Therefore, it becomes imperative to evaluate the performance of trained models on practical data captured under realistic conditions. Thus, in addition to analyzing the simulation results, some experiments were conducted to evaluate the performance of the ResNet-34 network on real-world data. This section presents the application of the trained network on the recorded dataset and discusses the classifier's performance.

To conduct the experiments, a quadrotor drone in both loaded and unloaded cases was deployed as the target (Figure 3.13). In addition, another quadrotor drone was equipped with a camera to capture images of drones at various distances (Figure 3.14). The air-to-air

setup of the test posed additional challenges due to the relative motion and perspective changes between the camera and the drones.

(a) Close view (b) Loaded drone in the sky

Figure 3.13: The target drone with the attached payload

All the tests were conducted by NRC. The observer drone was equipped with a camera with an FOV of 62.7 degrees. The captured frames had a resolution of 1080×1920 pixels and the camera recorded videos at 30 frames per second. The captured practical dataset encompasses a range of scenarios, with the quadrotor positioned at distances ranging from 20 to approximately 100 meters from the camera. This distance variation simulates real-world scenarios where drones may be present at different proximities to the observing camera. To simulate a realistic payload-carrying situation, a box-shaped payload measuring $7.8 \times 9 \times 15$ cm^3 was attached underneath the quadrotor. The drone occupied an average width of fewer than 20 pixels to more than 120 pixels in the frames. During the hovering phase of the observer drone, the camera captured videos of the target drone exhibiting different maneuvers. These maneuvers included approaches and retreats from the observer drone, lateral movements to capture various ground truth azimuth angles relative to the camera, as well as ascending and descending motions to cover a range of ground truth elevation angles. The flying path can be seen in Figure 3.15.

The simulated dataset offered a variety of backgrounds, including sky, trees, mountains, and buildings. However, this was not the case for the experimental data. The choice of the test field for the flight scenarios primarily resulted in capturing frames with

Figure 3.14: The observer drone with the mounted onboard camera

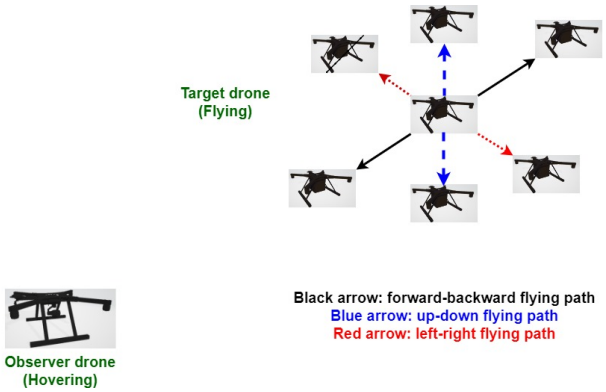
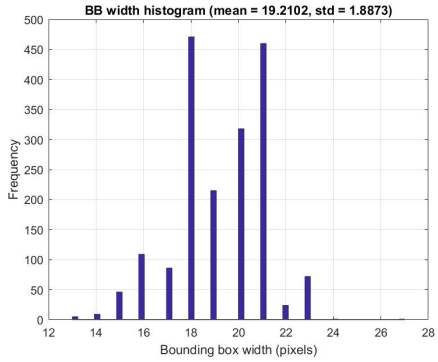


Figure 3.15: Observer and target drones' flying paths in the experimental test

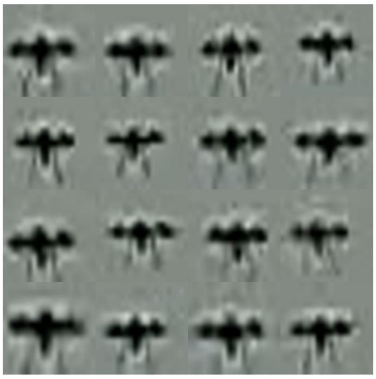
a sky background. Nevertheless, these backgrounds were captured under diverse lighting and weather conditions, ranging from sunny to cloudy. Additionally, as evident in the sample images presented in the following sections (e.g., Figure 3.19), tree backgrounds were also observed in certain frames. It is essential to note that these tree backgrounds were recorded under various light and weather conditions. Given this observation about the diversity of backgrounds in the real dataset, conducting flight tests in different environments with alternative backgrounds represents a potential area for future work. The deep networks employed in this study have exhibited promising results in addressing the drone payload detection problem using synthetic data with various backgrounds. This underscores the potential of the proposed methodology to perform effectively in scenarios with diverse backgrounds. Gathering real-world data in environments with different backgrounds would confirm whether the observed performance can be consistently achieved in practical applications.

The trained network was utilized to classify the recorded videos from loaded and un-

loaded drones. It could effectively solve the binary loaded vs. unloaded UAV classification problem with an accuracy from 85% to around 99% (Figure 3.16 to Figure 3.20) . These results clearly demonstrate the effectiveness of our simulated dataset in training the classifier network. The high accuracy attained on real-world data validates the generalization capability of the network, indicating its potential for practical deployment in similar scenarios. As an illustration, Figure 3.16 to Figure 3.20 showcase the specifications of the recorded data, along with the corresponding classification results. In each figure, the histogram of the width of the bounding box is presented. The width, represented in pixels, exhibits significant variability ranging from fewer than 20 pixels to higher values, reflecting the diverse sizes of the detected objects. Additionally, samples of the resized 255×255 images, which serve as the input to the ResNet network, are provided. These samples showcase the captured images from the target drone, revealing the challenges associated with variable backgrounds, varying lighting conditions, blurriness, and distortions. Despite these adverse factors, the classifier network demonstrates its capability to accurately classify the loaded vs. unloaded UAVs, thus showcasing its robustness in real-world scenarios.



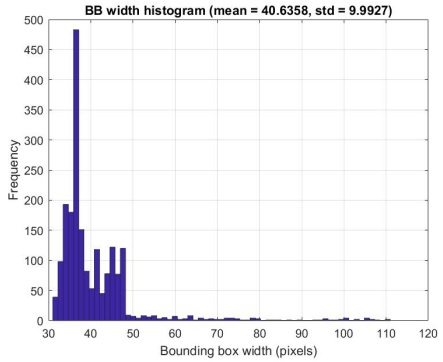
(a) BB width histogram



(b) Sample input images

Figure 3.16: Histogram of the width of bounding box and some sample input images to ResNet-34 (scenario 1: 2283 images, loaded drone, accuracy: 0.9995)

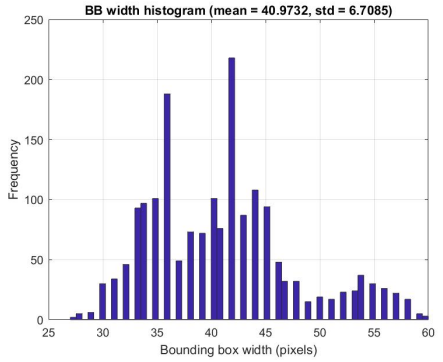
As evident in the sample images from the simulated dataset employed for training the ResNet-34 model, there is a Quadrotor present in the dataset that bears a similar shape (though not identical) to the drone utilized in the experimental test. To assess the model's performance when faced with a drone type it has not encountered before, the network was retrained using the same simulated dataset, with the exception of excluding the Quadrotor



(a) BB width histogram

(b) Sample input images

Figure 3.17: Histogram of the width of bounding box and some sample input images to ResNet-34 (scenario 2: 1974 images, unloaded drone, accuracy: 0.8919)

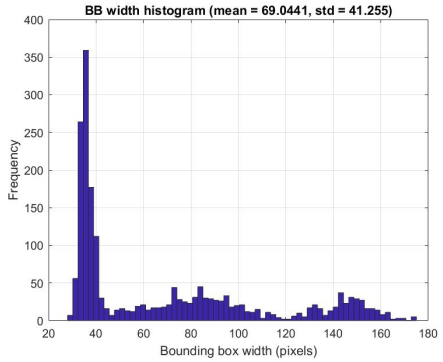


(a) BB width histogram

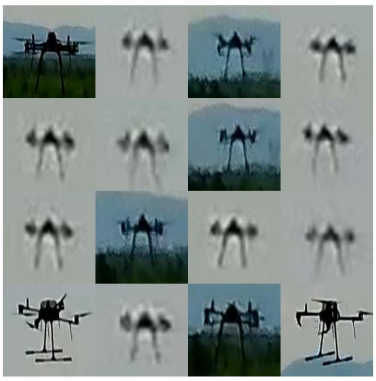
(b) Sample input images

Figure 3.18: Histogram of the width of bounding box and some sample input images to ResNet-34 (scenario 3: 1830 images, loaded drone, accuracy: 0.8728)

samples. The resulting retrained network was subsequently applied to the experimental data, as depicted in Table 3.3. These results demonstrate that the network trained using the synthetic data still retains the capability to address the binary classification problem of loaded versus unloaded drones. However, the numbers indicate superior performance in the case of loaded drones.

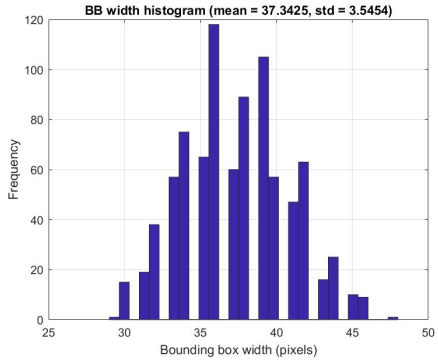


(a) BB width histogram

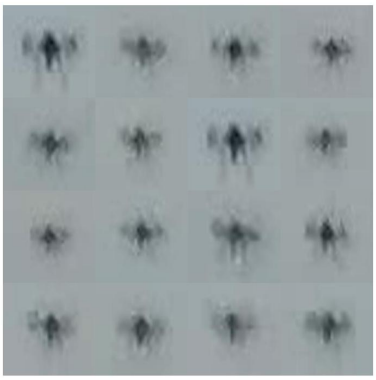


(b) Sample input images

Figure 3.19: Histogram of the width of bounding box and some sample input images to ResNet-34 (scenario 4: 2016 images, unloaded drone, accuracy: 0.8535)



(a) BB width histogram



(b) Sample input images

Figure 3.20: Histogram of the width of bounding box and some sample input images to ResNet-34 (scenario 5: 870 images, loaded drone, accuracy: 0.9776)

The same trained network can also be applied to the simulated dataset containing both loaded and unloaded Quadrotor models from AirSim©. In this case, the classification accuracy is 0.8924 for unloaded drones and 0.9431 for loaded drones. Notably, the simulated environment's conditions remain consistent between the training and testing phases, as the network is originally trained on simulated data. Consequently, even though the Quadrotor

model was not part of the training process, the network demonstrates high accuracy in classifying these cases. These findings suggest that with a sufficient amount of experimental data, the network can perform effectively even on previously unseen drone types.

Table 3.3: The effect of removing similar drone to the experimental one from the training simulated dataset on the classification accuracy of ResNet-34

Scenario	Accuracy
Scenario 1	0.7607
Scenario 2	0.6502
Scenario 3	0.9822
Scenario 4	0.6843
Scenario 5	0.9933

Another aspect that may raise concerns about the experimental test results is the state of the observer drone with the camera during data recording. As depicted in Figure 3.15, the observer drone was in a hovering position during the test, which meant the camera remained fixed with no movement from either the camera or the drone it was mounted on. This static condition may impact payload detection results (and subsequently, the localization analysis in the next chapter). Unfortunately, almost all the experimental flight tests with available GPS information, which could be used for both payload detection and localization analysis, were conducted with a hovering observer drone. However, among all the recorded data, there are a few tests in which the observer drone with the camera was also in flight during scene recording. Take, for example, the flight scenario illustrated in Figure 3.21. In this scenario, both the observer and target drones were in flight (not hovering). Figure 3.22 provides the histogram of the BB width and some resized sample images of the drone in this scenario. Notably, the ResNet-34 classifier managed to distinguish this drone as loaded with an accuracy of 0.8661, demonstrating its capability to address the payload detection problem even when the camera is in motion. The results of localization in this case will be presented in the next chapter.

3.5.2.1 Retraining with additional practical data

In order to further enhance the classification accuracy and demonstrate the effectiveness of the ResNet-34 network in addressing the problem, the network was retrained using a combination of the previously simulated data and additional practical data collected on a

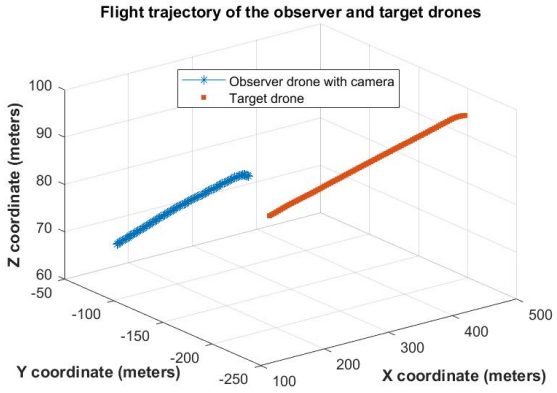
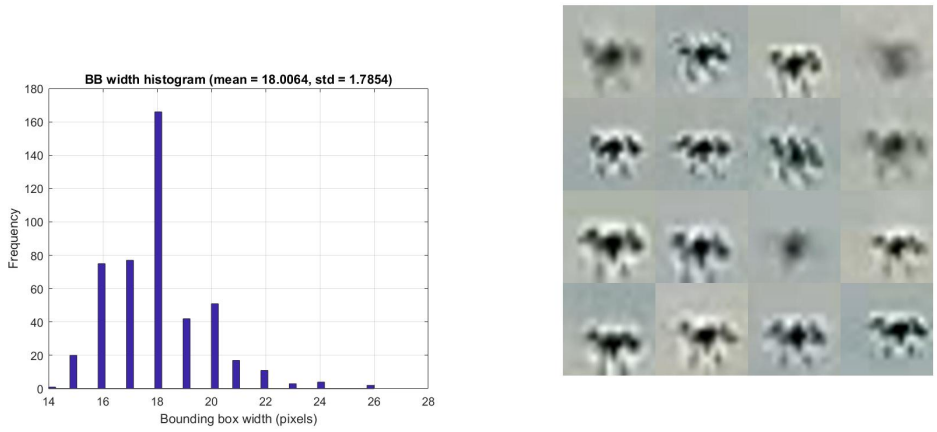


Figure 3.21: Flight path of the observer and target drones in a sample case of moving camera



(a) BB width histogram (b) Sample input images

Figure 3.22: Histogram of the width of bounding box and some sample input images to ResNet-34 (moving camera scenario, 469 images, loaded drone, accuracy: 0.8661)

different date and under different scenarios. While the performance of the network trained solely on simulated data was satisfactory, with an accuracy above 85%, the inclusion of practical data aimed to provide the network with exposure to real-world variations and improve its performance.

To augment the training data, an additional set of practical data was collected on a

different date and under different scenarios. This practical data, consisting of 1080 images per class, was added to the existing 11200 simulated images for both the loaded and unloaded cases. This addition amounts to approximately 20% of the total training data and serves to provide the network with exposure to real-world variations. It is important to note that while there are inherent differences between this added training data and the final test data, which is captured on a different date and in varying weather conditions, this inclusion helps the network become more familiar with similar images.

Figure 3.23 displays some examples of the additional experimental data that were included in the training set. These images, resized to 255×255 pixels, were used as input to the ResNet-34 network during the retraining process.

Figure 3.23: Samples of the additional experimental data collected from flight tests on a different date, used for training the ResNet-34 network.

By incorporating this practical data into the training process, the network gains exposure to real-world scenarios and increases its ability to accurately classify payload conditions, thereby improving its overall performance. For the previous 5 scenarios in Figure 3.16 to Figure 3.20, the results are given in Table 3.4. As can be seen, the accuracy improves by above 10% in some of the cases.

3.5.2.2 Applying other variants of ResNet and EfficientNet

The practical data analysis presented in Table 3.4 demonstrates the high accuracy achieved by the ResNet-34 network in detecting the payload of the drones. However, in scenarios where the complexity increases, employing a deeper network architecture may

Table 3.4: The effect of including experimental data in training the ResNet-34 network on the classification accuracy

Scenario	Acc. (simulated data training)	Acc. (mixed simulated/practical data training)
Scenario 1	0.9995	0.9996
Scenario 2	0.8919	0.9983
Scenario 3	0.8728	0.9978
Scenario 4	0.8535	0.9882
Scenario 5	0.9776	0.9977

prove beneficial for extracting more informative features to improve classification performance. One limitation of the previous work by Seidaliyeva et al. [5] is the availability of only 1000 frames for both training and validation, which led to the use of a shallower network like ResNet-34.

In this study, we have developed a simulated dataset comprising loaded and unloaded drones, which enables us to explore the potential of training deeper networks, such as ResNet-50 and ResNet-101. With this larger dataset and the promising results achieved with ResNet-34, employing deeper architectures can improve classification accuracy, particularly when dealing with more complex drone and payload shapes.

Table 3.5: Classification results (accuracy) on the experimental data using ResNet-50 and ResNet-101 architectures for the case of trained network using the simulated data

Scenario	ResNet-34	ResNet-50	ResNet-101
Scenario 1	0.9995	1.0	0.9996
Scenario 2	0.8919	0.9776	0.9983
Scenario 3	0.8728	0.9984	0.9984
Scenario 4	0.8535	0.9531	0.9765
Scenario 5	0.9776	0.9975	0.9989

The classification accuracy results for distinguishing between loaded and unloaded drones using the ResNet-50 and ResNet-101 architectures, compared to the baseline ResNet-34 model, are presented in Table 3.5. It is evident that both deeper networks achieved higher accuracy, ranging from a 2% to 12% improvement over the ResNet-34 model. This

improvement can be attributed to the more complex structure of the deeper networks, which enables them to extract more distinctive features and enhance accuracy. The availability of a large amount of data in the developed simulated dataset facilitated the training of these deeper networks.

In addition to the ResNet variants, we have also employed the EfficientNet-B2 classifier for addressing the drone payload detection problem. In the comparison of ResNet-34 and EfficientNet-B2 for drone payload detection using a network trained with simulated data, EfficientNet-B2 consistently outperforms ResNet-34 across various scenarios (Table 3.6). In all sample scenarios, EfficientNet-B2 exhibited better performance, with accuracy ranging from around 94% to 100%, while ResNet-34 scored lower, from 85% to 99%. This suggests that EfficientNet-B2 is a superior choice for the drone payload detection problem, as it offers improved accuracy over ResNet-34 with lower computational load.

Table 3.6: Classification results (accuracy) on the experimental data using EfficientNet-B2 architecture for the case of trained network using the simulated data

Scenario	ResNet-34	EfficientNet-B2
Scenario 1	0.9995	1.0
Scenario 2	0.8919	0.9672
Scenario 3	0.8728	0.9534
Scenario 4	0.8535	0.9489
Scenario 5	0.9776	0.9985

3.6 Summary

In conclusion, this chapter focused on the problem of drone payload detection and presented a method based on the ResNet and EfficientNet architectures for distinguishing between loaded and unloaded drones. The chapter first defined the problem and highlighted its significance in various applications. It then explained the ResNet-34 architecture (and other variants of ResNet architecture including the ones with 50 and 101 layers) and its innovative approach of residual learning, which enables training deeper networks without sacrificing accuracy. Furthermore, the chapter explained EfficientNet as a newer classifier, highlighting the selection of its B2 variant due to its superior performance and lower computational load compared to ResNet. The chapter also discussed the preparation of a

simulated dataset specifically designed for drone payload detection, including the importing of custom drone models into the AirSim© environment and the simulation of loaded drones and diverse environments.

The ResNet-34 network was trained with the simulated dataset, where the input images were cropped around the detected bounding boxes and resized to a fixed size. The developed dataset included different types of drones, simulated payloads, and various backgrounds, providing diversity and complexity for training and evaluation. The statistics of the dataset were presented, showcasing the distribution of classes, spatial locations of drones, and the sizes of bounding boxes. The trained network demonstrated promising potential for accurately classifying loaded and unloaded drones for both simulated and experimental datasets.

Finally, to enhance the classification accuracy and demonstrate the effectiveness of the ResNet-34 network, it was retrained using a combination of simulated data and additional practical data collected under different scenarios. The inclusion of practical data, which accounted for about 20% of the training data, aimed to expose the network to real-world variations and improve its performance. The results showed a significant improvement in accuracy, with an increase of over 10% in some scenarios when using the mixed simulated and practical data for training compared to training solely on simulated data. Additionally, the new simulated dataset with a huge amount of data provides this opportunity to train the ResNet-50 and ResNet-101 networks, and also EfficientNet-B2, for addressing the loaded vs. unloaded drone classification problem. Both ResNet networks could increase the classification accuracy compared to the base architecture (ResNet-34) due to their more complex structures which enables them to extract more informative features from the image. Additionally, EfficientNet-B2 outperformed ResNet-34 while maintaining lower complexity, making it a compelling choice for real-time applications.

Chapter 4

Drone localization

4.1 Introduction

Drone localization is essential for enabling autonomous navigation, obstacle avoidance, and efficient mission planning. This chapter focuses on exploring different approaches for drone localization using captured images/videos from it. Specifically, we examine the utilization of visual-based methods to extract 3D location information of the observed drone. Visual-based methods leverage fixed cameras on the ground or cameras onboard the observer drone to extract visual features from the environment and estimate the target drone's position. In this chapter, we begin by discussing classical and DL-based approaches for 3D localization using 2D images. Next, for utilizing localization measurements from different sensors, a data-preparation algorithm is developed. Subsequently, the performance of the localization methods using both simulated and experimental data is analyzed. Lastly, the developed counter-UAV multi-sensor simulator will be discussed..

4.2 3D localization using 2D image

After the initial step of detecting a drone in the captured image or video using vision object detection tools such as YOLO, the user is provided with a bounding box that encompasses the identified object, specifically the drone. This bounding box contains valuable information about the size and position of the drone within the image frame, as well as the corresponding image area that represents the drone.

The main challenge at hand is to leverage this bounding box information to accurately estimate the three-dimensional (3D) location of the drone within the camera's coordinate frame. This task requires determining the drone's position in terms of its distance from the camera, as well as its azimuth and elevation angles with respect to a coordinate frame where the camera is positioned at the origin.

To tackle this challenge, two distinct approaches can be employed: the classical approach and the DL-based approach. In the classical approach, geometric principles and computer vision techniques are utilized to extract relevant features from the captured image, such as the size and position of the bounding box. These features are then used in conjunction with camera parameters including the focal length, image resolution, and zoom condition, to calculate the drone's 3D location. On the other hand, the DL-based approach leverages the power of deep learning algorithms and neural networks to directly estimate the 3D location of the drone from the captured image. Through the use of annotated training data, the deep learning model learns to extract relevant spatial features from the bounding box and image context, enabling it to predict the drone's distance.

In the following subsections, each of these approaches will be elaborated upon in more detail, highlighting their underlying methodologies, advantages, and limitations.

4.2.1 Classical approach

The geometry of the camera and the process of capturing images inherently result in variations in the size of objects depicted within the image, proportional to their distance from the camera. This phenomenon can be attributed to the principles of perspective projection and geometric transformations involved in the image formation process.

When capturing an image, the camera acts as the projection device, mimicking the human visual system. The perspective projection introduces a foreshortening effect, where objects farther away from the camera appear smaller in the image compared to their actual size. This is due to the decrease in the angular size of objects as the distance between them and the camera increases. To elaborate further, consider an object positioned at varying distances from the camera. As the object moves away from the camera, the captured image portrays it with a decreasing size relative to its actual dimensions. Conversely, as the object approaches the camera, its size within the image increases accordingly. This size-distance relationship forms the basis for estimating the object's distance from the camera, as it provides a direct geometric correlation between the two.

Also, the position of an object within the captured image can be exploited to estimate its azimuth and elevation angles with respect to a coordinate frame, where the camera is

situated at the origin. This process involves mapping the object's position in the image to its corresponding position in the 3D world space.

Although achieving highly accurate estimations of distance and azimuth/elevation angles typically requires precise knowledge of the camera's intrinsic parameters, such as focal length, principal point, and lens distortion coefficients, the estimation procedure can be simplified to some extent for the specific context of this research project. This simplification comes at the cost of increased estimation error.

In this research project, the focus is on drones flying at distances less than 100 meters from the camera. At such distances, it is anticipated, based on the experimental data, that certain factors such as lens distortion have not that much impact on the results (or at least, do not worsen the results beyond the expected performance). Therefore, for the purposes of this project, the effects of lens distortion can be considered negligible.

Furthermore, it is important to acknowledge that the classical approach employed in this research may not yield highly precise results. The expectation is that the estimation of distance may have an error tolerance similar to that of GPS accuracy. While striving for accuracy, it is recognized that achieving pinpoint precision is not essential for the goals of this study.

Given these assumptions and considerations, the procedure for estimating the distance and azimuth/elevation angles can be outlined as follows.

Firstly, the focal length of the camera can be estimated by utilizing a reference case through (4.1):

$$\hat{f} = \frac{P_r \times D_r}{W_r} \tag{4.1}$$

The values are assumed to be measured in the reference case. In this equation, P_r represents the number of pixels occupied by the width of the object, D_r corresponds to the distance of the object from the camera, and W_r denotes the actual width of the object in meters. This estimation provides an approximation of the focal length based on the relationship between pixel measurements and physical dimensions. After the estimation of the camera's focal length, the distance for a test case can be estimated using Equation (4.2), which considers the number of occupied pixels (P_t) and the width of the object (W_t) in meters:

$$\hat{D}_t = \frac{W_t \times \hat{f}}{P_t} \tag{4.2}$$

This equation allows us to calculate the estimated distance of the object in the test

case. By knowing the actual width of both the reference case (W_r) and the test case (W_t), as well as the number of occupied pixels in each case (P_r and P_t), the distance estimation becomes feasible. It is worth noting that the reference case is used only once to estimate the focal length ($\hat{f}$), and subsequently, this value can be utilized for any test case.

In our specific scenario, the number of pixels in the test case corresponds to the width of the bounding box obtained from the object detection algorithm, such as YOLO. This approach allows us to leverage the captured image's information to estimate the distance of the object. Additionally, when the widths of the reference case (W_r) and the test case (W_t) are equal, the ratio between the number of pixels in the reference case and the number of pixels in the test case would be inversely proportional to the ratio of the distances. This means that as the number of pixels decreases in the test case compared to the reference case, the distance to the object increases accordingly.

This relationship is valuable in scenarios where the exact width of the object is not known but remains constant between the reference and test cases. By comparing the number of pixels in the bounding boxes, it is possible to estimate the relative distance of the object without explicitly determining its width. However, it is important to note that this method assumes a consistent object size and may introduce additional errors if the width of the object varies significantly between the reference and test cases.

Finally, it is important to consider a drawback of this method, namely the necessity to know the actual width of the object. The accuracy of the distance estimation relies on accurately determining the object's width. If the width is not known precisely or if there are variations in the width due to different orientations or deformations of the object, it can introduce errors in the distance estimation. Thus, the estimation process depends on obtaining reliable width information for accurate results.

In addition to estimating the distance, one can approximately estimate the azimuth and elevation angles of the drone based on its position in the captured image. By leveraging the camera's FOV and resolution, the planar region within the FOV can be assumed, enabling the estimation of azimuth and elevation angles using Equation (4.3):

$$\begin{cases} \hat{\theta} \approx \frac{\text{pixel}_{ver}}{N_{ver}} \times \text{FOV}_{ver}(rad) \\ \hat{\varphi} \approx \frac{\text{pixel}_{hor}}{N_{hor}} \times \text{FOV}_{hor}(rad) \end{cases} \tag{4.3}$$

Here, N_{hor} and N_{ver} represent the total number of pixels in the image frame in the horizontal and vertical directions, respectively. The FOV in both directions is denoted by FOV_{hor} and FOV_{ver}. The variables $pixel_{hor}$ and $pixel_{ver}$ correspond to the drone's position in the pixel index within the image frame. These values are obtained from the

object detection method, such as the YOLO algorithm, which identifies and localizes the drone in the image. In this thesis, the position of the drone within the image frame is determined by considering the center of the bounding box. Extracting other features, such as the center of mass, solely from the recorded vision data, is not a straightforward task for this purpose. The concept of this estimation method is illustrated in Figure 4.1. Notably, these approximate equations demonstrate satisfactory results for angle estimation within the range of distances considered in the experimental tests, as demonstrated in the simulation section.

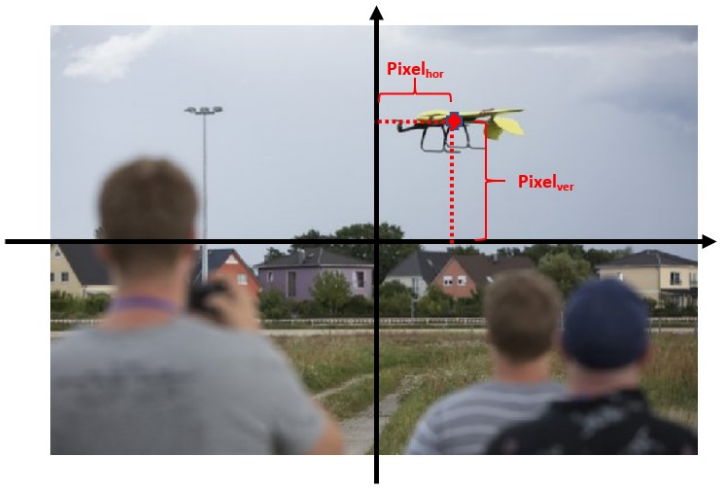


Figure 4.1: Simple model for elevation and azimuth angle estimation in the image

The presented method offers several advantages in estimating azimuth and elevation angles. Firstly, it relies on readily available information, such as the camera's FOV and resolution, as well as the drone's position within the image frame, which can be obtained through object detection techniques. This simplifies the estimation process and reduces the reliance on complex calibration procedures or additional sensors. Moreover, the method is computationally efficient, allowing for real-time or near real-time estimation of the drone's angles. Additionally, the method is applicable to various camera setups and can be easily adapted to different scenarios and environments.

However, there are certain drawbacks to consider. One limitation is the assumption of a planar region within the camera's FOV, which may introduce errors when the drone deviates significantly from this assumption or when capturing scenes with complex geometries. Additionally, the accuracy of the angle estimation is influenced by factors such as the resolution of the image, the precision of the object detection algorithm, and any potential

image distortions. Finally, it is important to note that the estimation of azimuth and elevation angles using this method is approximate and may not provide the same level of accuracy as more sophisticated techniques or additional sensor data.

Despite these limitations, this method offers a practical and effective means of estimating azimuth and elevation angles based on readily available image data. It provides a valuable tool for 3D localization using 2D images, especially in scenarios where a high level of accuracy is not the primary requirement and where simplicity, real-time processing, and cost-effectiveness are prioritized.

In regards to the classical approach, it is important to acknowledge that while the estimation results may not be highly accurate and may exhibit some level of noise with a bounded error, there is a key consideration to be aware of. This consideration revolves around the inherent limitations of the drone's dynamic structure, which restrict its ability to perform rapid and drastic maneuvers. Consequently, it is reasonable to expect that the deviations observed in the estimated location parameters will remain within certain bounds.

For instance, let us consider a scenario where the camera operates at a frame rate of 30 frames per second (similar to the camera used in the experimental test discussed in section Section 4.4.3). If the distance estimation output suddenly jumps from 70 to 80 meters between consecutive frames, it would imply that the drone would need to exhibit a velocity of approximately 300 meters per second, which is not realistically achievable. This observation allows us to exploit the nature of the drone's trajectory. By recognizing that the output of this method represents the trajectory of a flying drone, we can apply appropriate tracking (or smoothing) filters to mitigate the effects of deviations and noise [65–67]. For example, techniques like the Kalman filter, or even a simple Moving Average (MA) filter, as demonstrated in the results section, can effectively smooth out these deviations and provide more reliable and refined estimations. These filtering techniques help in achieving a more consistent and coherent trajectory by effectively reducing the impact of sudden and unrealistic changes in the estimated location parameters. They leverage the assumption that the drone's movement is subject to physical constraints and cannot exhibit abrupt and extreme variations within short time intervals. Applying tracking filters not only improves the overall accuracy of the estimations but also enhances the interpretability of the obtained trajectory, making it more reliable and suitable for further analysis and applications.

However, it is important to note that the choice of the specific tracking filter depends on various factors, including the characteristics of the drone's motion, the level of noise present in the estimation results, and the computational resources available. Therefore,

careful consideration should be given to selecting the most appropriate filtering technique based on the specific requirements and constraints of the application.

In summary, while the classical approach may yield estimation results with some degree of noise and bounded error, leveraging the drone's dynamic limitations and applying tracking filters can effectively mitigate these challenges and provide smoother and more accurate trajectory estimations.

4.2.2 DL-based approach

In this section, we explore the use of a regression approach with the designed CNN for estimating the distance based on cropped images obtained from the bounding box output of an object detector network, specifically YOLOv7. Using regression allows us to estimate the distance based on visual features captured in the cropped images [68, 69]. By training the network on a dataset with ground truth distance labels, the network can learn to associate visual features with varying distances, enabling it to make distance predictions based on the observed patterns. In many scenarios, there exists a correlation between visual features and the distance between the camera and the drone. As the drone moves closer or farther away, visual cues such as size, perspective, or sharpness can change. By training a regression network on a diverse dataset that captures these visual cues, the network can learn to exploit these correlations and estimate the distance accurately. Also, the relationship between visual features and distance is often non-linear. A regression approach with deep neural networks, such as the provided CNN architecture, can capture complex non-linear mappings between the input image features and the distance. The convolutional layers enable the network to learn hierarchical representations, extracting relevant features at different scales and orientations, which can be beneficial for capturing non-linear relationships. Finally, utilizing a regression approach can be computationally efficient compared to other methods that require explicit 3D geometry calculations or complex multi-view analysis. Once trained, the regression network can make distance predictions in real-time by simply processing the cropped images, making it suitable for applications that require quick and continuous distance estimations.

It is important to note that the success of this approach depends on the quality and diversity of the training dataset, as well as the ability of the network to learn relevant features that correlate with the distance between the camera and the drone. Additionally, the network's performance may be affected by factors such as lighting conditions, occlusions, and variations in the appearance of drones. Regular monitoring, evaluation, and potential fine-tuning may be necessary to ensure accurate distance estimation in different scenarios.

4.2.2.1 Regressor architecture

The regressor network can be seen in Figure 4.2. The first layer (imageInputLayer)

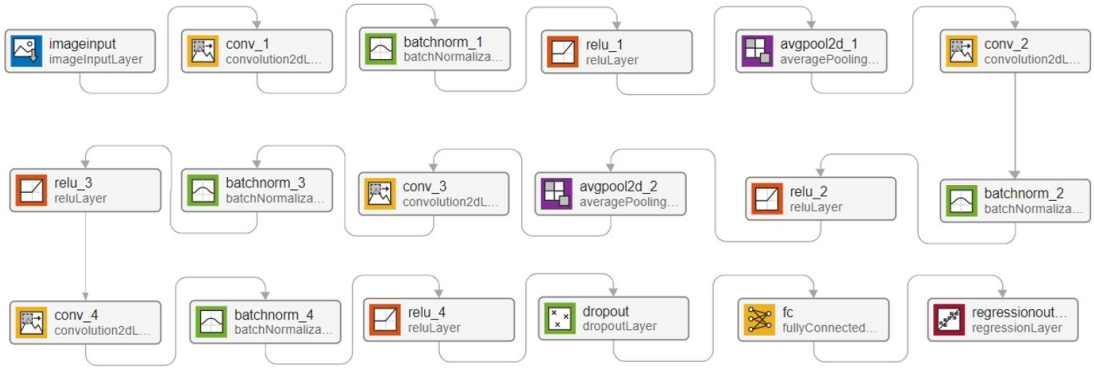


Figure 4.2: The architecture of the CNN-based distance regressor

defines the input size of the network, which is based on the size of the training images (XTrain). It ensures that the network expects input images with the same dimensions. In this study, the size of input images is assumed to be 80×150 pixels (number of pixels in the vertical and horizontal axis, respectively). The image input to this network is a cropped image of the bounding box area generated by the object detection network. In cases where the bounding box size is different from 80×150 pixels, the cropped image needs to be resized to match these dimensions. The resizing process employs the Bicubic interpolation method. It is important to note that all the images are assumed to be in the compressed *JPEG* format. The first convolutional layer is determined to have 8 filters of size 3x3. The next layer is the batch normalization layer. By performing batch normalization which normalizes the activations of the previous layer, the stability and speed of training improve. The next ReLU layer introduces non-linearity and captures complex patterns in the data. At the next layer, the pooling operation performs 2×2 average pooling with a stride of 2, reducing the spatial dimensions of the feature maps by half. These layers are repeated twice, progressively increasing the number of filters (16 in the second convolutional layer and 32 in the third and fourth convolutional layers). The next layer applies dropout regularization, randomly setting 20% of the activations to zero during training. It helps prevent overfitting and improves generalization. The fully connected layer establishes connections between all the output activations from the previous layer, ultimately reducing the output to a single scalar value. It serves as the regression head of the network. Finally, the regression layer computes the loss between the

predicted distance and the true distance. It is specifically designed for regression tasks and ensures that the network is trained to minimize the regression error.

CNN architecture is well-suited for image-based tasks due to its ability to capture spatial hierarchies and learn local patterns. This architecture takes advantage of the convolutional layers to extract relevant features from the UAV images. In each level of the architecture, the inclusion of a batch normalization layer helps normalize the activations, which can improve the training process by reducing internal covariate shifts and accelerating convergence. The ReLU activation function introduces non-linearity, enabling the network to learn complex representations and capture intricate relationships within the data. Also, the average pooling layers help downsample the feature maps and capture the most salient information in an aggregated form.

The above network structure does not have an extremely deep architecture, which could limit its ability to capture more intricate and hierarchical features. However, as demonstrated in the simulation results, the selected network, despite its simple structure and low complexity compared to deeper networks, exhibits acceptable performance in distance estimation, achieving low absolute error. Regarding the angle estimation, one can still rely on the same approach discussed in the classical section (Section 4.2.1).

It is worth mentioning that drone localization is performed frame by frame, similar to the payload detection algorithm. In other words, even for false detections from the detector network, there will be reported locations from the localization block. As previously explained in Section 3.3, certain post-processing algorithms should be employed on the output of the localization block to mitigate such inaccuracies. Solutions like activating the localization network only for targets with high confidence can also be employed in this context. Developing methods to address these scenarios could be considered for future research.

4.3 Localization using various sensors

In many scenarios, particularly in the area of drone detection and localization, it is common to deploy a combination of sensors to enhance situational awareness. One such combination involves the use of radar and cameras, including both stationary on-ground cameras and cameras mounted on observer drones. To effectively combine the measurements from these sensors using fusion techniques, it is often necessary to transfer all the measurements to a single reference coordinate frame, ensuring compatibility for fusion. However, achieving data fusion is not a straightforward process. Once the measurements

are translated into the reference frame, there remains a challenge of associating the corresponding measurements across the various sensors. This data association task involves matching observations that correspond to the same physical target. Moreover, a key aspect of sensor data fusion lies in selecting appropriate fusion techniques that maximize the utilization of information extracted from the individual sensors.

One of the main motivations for utilizing multiple sensors stems from the inherent strengths and capabilities that each sensor may have in providing certain parameters more accurately than others. For instance, when detecting drones using radar and PTZ cameras, radar offers more precise range estimation, while PTZ cameras deliver better angle measurements.

In this section, we focus on addressing the specific challenge of using a stationary radar and one or more moving cameras, mounted on observer drones, to monitor the environment and detect intruder drones. An algorithm is developed that facilitates the translation of all the measurements to a shared coordinate frame (without loss of generality, here the radar coordinate frame is selected as the final frame). The output of this algorithm can serve as input for any fusion technique that aims to combine the information from diverse sensors effectively.

4.3.1 Problem definition

Assume that multiple sensors with distinct capabilities are available in a surveillance scenario:

- Stationary Sensor (Radar): there is a stationary radar sensor that is capable of detecting moving objects. It provides range measurements, as well as azimuth and elevation angles of the drone.
- Moving Sensor (Camera on UAV): There are one or more moving sensors represented by cameras mounted on UAV. These sensors have positional information either through GPS or by being located using the same radar system as mentioned earlier. The images captured by the cameras are assumed to be used to measure the distance of the target drone with respect to the origin of their internal coordinate frame (for instance, by using the classical or DL-based method mentioned in the previous sections). In addition to the GPS sensor, it is assumed that there is a Gyro mounted on the UAV which provides the roll/pitch/yaw angles. Also, the camera is assumed to be mounted on a fixed-gimbal, otherwise, the rotation angles of the gimbal need to be provided.

To fuse the measurements of these sensors regarding the localization of the target drone, it is needed to develop an algorithm that can effectively convert all distance measurements to a consistent coordinate system, preferably the radar's coordinate system. This conversion ensures that the measurements obtained from different sensors can be seamlessly integrated and fused for further analysis and decision-making purposes.

4.3.2 Developed algorithm

To solve the problem, consider the case of one radar and only one camera mounted on the observer drone (the generalization to the case of several cameras is straightforward). Based on the problem definition, the radar provides measurements of the target's range, elevation angle, and azimuth angle with certain degrees of uncertainty, all relative to the radar's location. Furthermore, it is assumed that the GPS data of the second sensor's location, including latitude, longitude, and elevation values, is already known. All the coordinates should be translated to the radar frame, as depicted in Figure 4.3 (although this can be easily adjusted to any other fixed coordinate system as the reference).

Considering Figure 4.4, the following steps should be done to solve the problem:

1. **Converting the radar's spherical coordinate measurements to the Cartesian system, accounting for uncertainties:** It is important to note that converting the measurements from spherical coordinates to the Cartesian system is a non-linear transformation. In order to handle the uncertainties associated with these measurements, a linearization technique based on the first-order Taylor series approximation can be employed. However, it is worth mentioning that this linearization process may introduce biases in the estimation errors. Alternatively, more advanced methods such as the *unscented transformation* proposed by Julier and Uhlmann [70] can be utilized for improved accuracy and reliability. In the current step, a simple linearization approach has been adopted, but it can be refined in subsequent stages if necessary. Referring to Figure 4.5, the conversion process can be described as equation (4.4):

$$\begin{cases} x = \rho sin\theta cos\varphi \\ y = \rho sin\theta sin\varphi \\ z = \rho cos\theta \end{cases} \tag{4.4}$$

   By using Taylor series expansion, and only considering the first order approximation, the mean and covariance of the Cartesian coordinates after transformation will be as

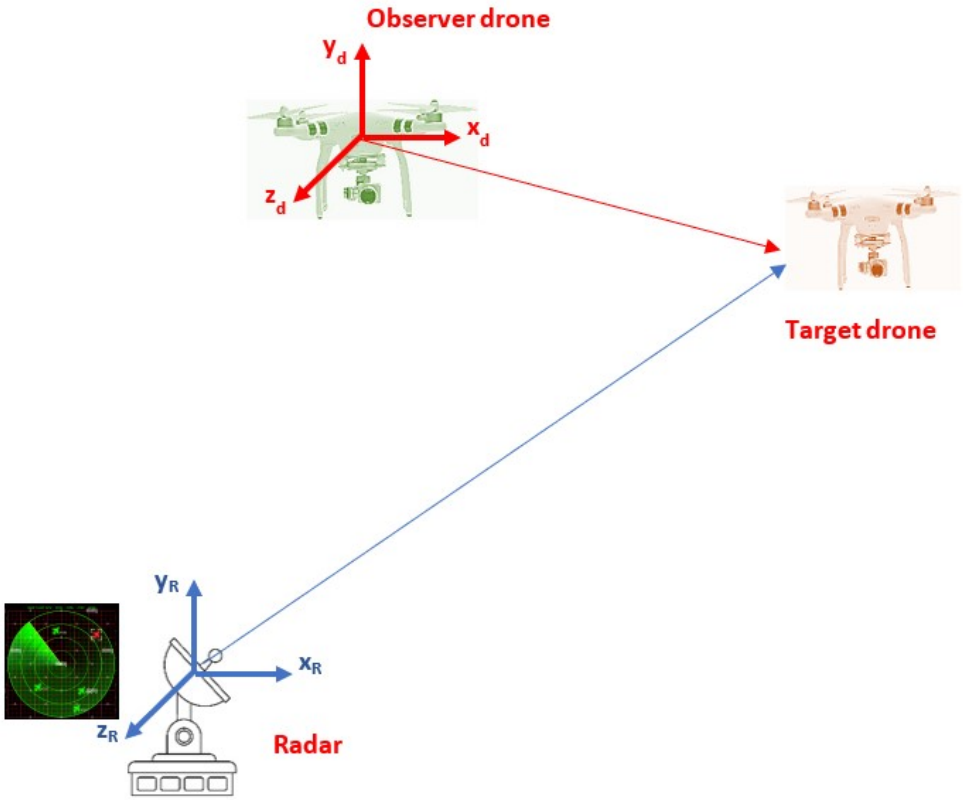


Figure 4.3: Considered scenario for localization using various sensor

follows:

$$\begin{cases} x = \overline{\rho} sin\overline{\theta} cos\overline{\varphi} \\ y = \overline{\rho} sin\overline{\theta} sin\overline{\varphi} \\ z = \overline{\rho} cos\overline{\theta} \end{cases} \qquad \mathbf{\Sigma}_{cart} = J\mathbf{\Sigma}_{sph}J^T \tag{4.5}$$

where Σ_{cart} and Σ_{sph} show the covariance of the variables in the Cartesian and spherical systems, respectively, and J is the Jacobian matrix. Considering uncorrelated uncertainties in the spherical coordinate system, we have:

$$\Sigma_{\mathbf{sph}} = \mathbf{diag}\left(\sigma_\rho^2, \sigma_\theta^2, \sigma_\varphi^2\right), \; J = \begin{bmatrix} sin\theta cos\varphi & \rho cos\theta cos\varphi & -\rho sin\theta sin\varphi \\ sin\theta sin\varphi & \rho cos\theta sin\varphi & \rho sin\theta cos\varphi \\ cos\theta & -\rho sin\theta & 0 \end{bmatrix} \tag{4.6}$$

2. **Assuming plane form for the scene in the camera and estimating the eleva-**

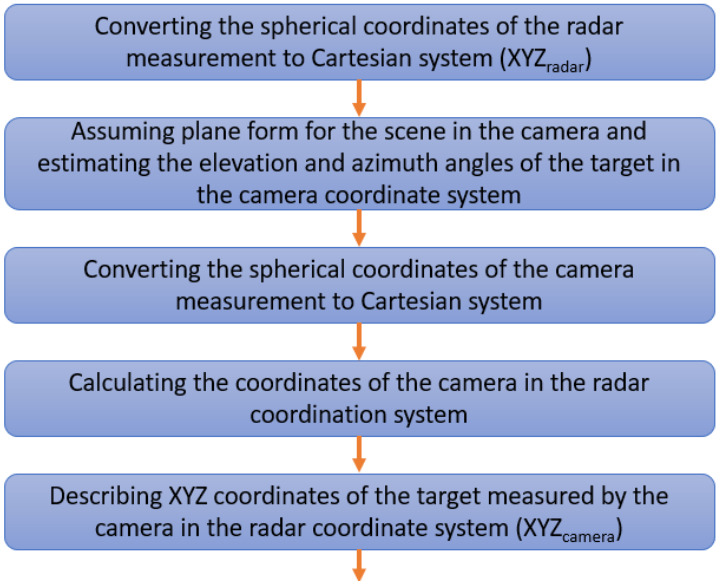


Figure 4.4: The steps for the coordinate conversion in the multisensor case

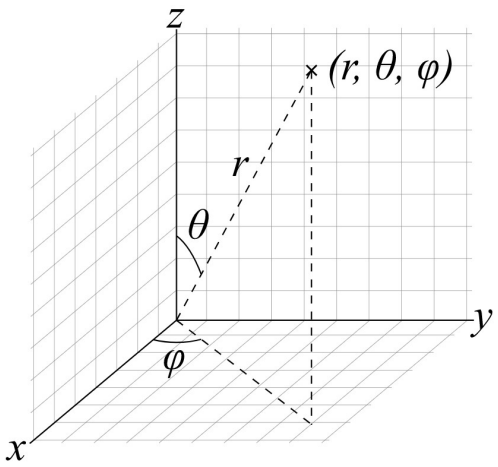


Figure 4.5: Spherical and Cartesian coordinates

tion and azimuth angles (θ and φ of the spherical coordinate system of the camera): to obtain the 3D coordinates of the target drone within the camera coordinate frame, it is necessary to estimate the corresponding angles. The approximate equations for performing these calculations are provided in equation (4.3).

3. **Converting the spherical coordinates of the camera measurement to Cartesian system including the uncertainties:** using similar equations in step 1, the spherical coordinate of the object in the camera image can be converted to the Cartesian coordinate in the camera coordinate system.

4. **Calculating the coordinates of the camera in the radar coordination system:** to calculate the coordinates of the camera in the radar's coordinate system, we can use the available GPS data for both the camera and the radar. By applying equation (4.7), we can derive the XYZ coordinates of the camera relative to the radar. Assuming the GPS information for the radar is available and for small distances between the observer drone and the radar, the Cartesian distances between these two points are as follows [71]:

$$\begin{cases} \Delta x = [R + (A_2 + A_1)/2](\varphi_2 - \varphi_1) * cos[(\theta_2 + \theta_1)/2] \\ \Delta y = [R + (A_2 + A_1)/2](\theta_2 - \theta_1) \\ \Delta z = A_2 - A1 \end{cases} \tag{4.7}$$

   In this equation, θ_k, φ_k, and A_k are the latitude, longitude, and altitude of the kth sensor. The latitude and longitude should be substituted in radians. The variable R also shows the base radius of the earth which is around 6371 kilometers. It is important to note that these equations may yield some inaccuracies and can be replaced with more accurate alternatives. Finally, in MATLAB, this conversion can be performed using the command *latlon2local*.

5. **Final transformation:** till now, the coordinates of the detected object in radar coordinate system (XYZ_{radar} from step 1), the coordinates of the detected object in camera coordinate system (XYZ_{camera} from step 3) and the coordinates of the camera in the radar coordinate system ($XYZ_{camera-in-radarsys}$ from step 4) are calculated. In addition to this information, we also need the roll, pitch and yaw angles of the camera with respect to the initial state of its coordinate system. In other words, assume that before starting flying, the direction of XYZ axes of both coordinate systems (camera and radar) is the same (and only a transformation exists between their origin). At the time that the object is detected, the roll, pitch, and yaw angles of the camera coordinate system with respect to these initial directions should be determined. To achieve the final conversion of the object's coordinates from the camera's coordinate system (XYZ_{camera}) to the radar's coordinate system, the equations presented in Figure 4.6 can be employed. These equations facilitate the necessary transformation and alignment between the camera and radar coordi-

nate systems. A series of operations including one translation and three rotations are required in this transformation.

In Figure 4.6, the black reference frame represents the radar's coordinate system, whereas the colored plots depict the camera's coordinate system. In reality, there is only one frame with rotations around the three axes compared to the reference frame. However, to clearly illustrate these rotations, three different frames have been plotted. The roll, pitch, and yaw angles are represented by θ, φ, and ψ, respectively. These angles define the rotational adjustments needed to align the camera's measurements with the radar's reference frame. The XYZ_T values correspond to the camera's position in the radar's coordinate frame, calculated in step 4. These values indicate the translation required to establish the spatial relationship between the camera and the radar. Finally, the XYZ' values represent the Cartesian coordinates of the detected target, measured in the camera's coordinate frame. These coordinates are the outcomes of step 3, reflecting the object's position as observed by the camera.

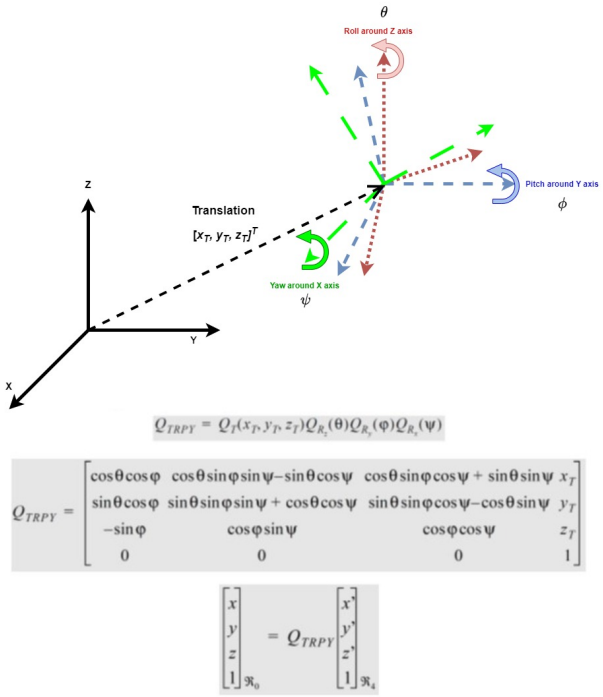


Figure 4.6: Transformation of object XYZ values from the camera coordinate system to the radar frame [8]

.

Before concluding this section, it is important to highlight some key aspects of the method discussed above. As indicated in the list of assumptions in Section 4.3.1, we assumed that the gimbal remains in a fixed position. Throughout the previous paragraphs, we have worked with the assumption of a single coordinate frame on the observer drone, which is also used for the camera. In the case of a fixed gimbal, where the camera's rotations are directly coupled with the drone's movements, this assumption introduces only a negligible error due to the differences between the camera's and drone's frames. In other words, the fixed gimbal setup leads to a slight rotation difference between these two frames. Additionally, there is a translation between these frames caused by the distance between the camera and the center of the drone's frame (the assumed position of mounting the GPS sensor). While this frame difference does introduce a small error, for greater accuracy, it is possible to compensate for this effect during the steps outlined in Figure 4.4. In this case, before the final two steps, the measurements in the camera frame should be initially converted into the drone's frame, using a similar conversion method as explained in Figure 4.6 but with parameters specific to the camera's and the drone's frames. The last two steps of Figure 4.4 would then involve the conversion between the drone's and the radar's frames.

However, when dealing with a moving gimbal, this error is no longer negligible, and the conversion between the camera's and drone's frames before the last two steps of Figure 4.4 should always be applied. The key distinction between the fixed gimbal case and the moving gimbal case is that in the former, this conversion is based on constant parameters. In contrast, for the moving gimbal scenario, the gimbal's rotation angles must be continuously monitored and utilized in the coordinate conversion.

4.4 Results

In order to evaluate the effectiveness of the developed algorithms for distance estimation using 2D images, it is necessary to have a dataset with known distance values as GT data. This allows for a comparison between the results obtained from the localization algorithm and the GT values. This section presents the analysis of the algorithm's performance using three types of data.

- **Simulated Data**: The use of simulated data provides a controlled environment for evaluating the algorithms' performance. AirSim® simulator provides the capability to save the Cartesian coordinates (XYZ values) of all the drones in the internal fixed coordinate frame. When a camera is mounted on one drone, serving as the observer,

and another drone is selected as the target, the simulator provides the necessary information to extract GT values for the distance, azimuth, and elevation angles between the camera and the target drone. These GT values can be easily calculated using a simple Cartesian-to-spherical transformation equation.

- **Fixed Camera and Drone on the Ground**: To obtain more realistic data, a test setup is designed in this study to accurately measure the distance between the camera and the grounded drone, which can be used as the GT value. Civil engineers commonly employ TSS equipment for precise distance measurements, achieving accuracies in the millimeter range. Leveraging this equipment, a test setup is designed to collect data. In this setup, the camera used for capturing video from the UAV is fixed onto the TSS. The UAV is positioned in close proximity to the prism of the TSS system. As a result, the distance between the TSS's main station and the prism closely approximates the distance between the camera and the UAV, with millimeter accuracy. This information can be utilized as the GT distance for the estimation process. Figure 4.7 illustrates the concept of utilizing this setup.

Figure 4.7: Concept of using TSS equipment for measuring the GT distance between the camera and the target drone.

- **Experimental Flight Test**: While the previous two setups provide initial datasets for analyzing the performance of the localization methods, the final performance evaluation should be conducted using a real flight scenario where both the drone

with the mounted camera and the target drone are flying. In this scenario, GT values can be extracted from GPS sensors mounted on both drones. By obtaining latitude, longitude, and altitude readings for both drones, it is possible to calculate the range, azimuth, and elevation angles between the camera and the target drone. In this study, it is assumed that the location information obtained from GPS data is error-free and serves as a reliable ground truth reference. However, as a direction for future research, it is recommended to consider the inherent uncertainties associated with GPS measurements when utilizing this data source.

By utilizing these three types of data sources for generating GT values, the performance of the localization algorithms can be thoroughly evaluated and compared against the ground truth. The simulated data allows for controlled testing, while the fixed camera and drone setup on the ground adds a higher degree of realism. Finally, the experimental flight test offers valuable insights into the algorithms' performance in real-world scenarios, thus complementing the evaluations conducted using simulated and fixed camera and drone setups on the ground. The utilization of these three types of data sources provides a comprehensive understanding of the reliability and accuracy of the distance estimation approach.

4.4.1 Simulated data

This section focuses on analyzing the performance of the proposed localization methods using the simulated data from AirSim©. As discussed in the section Section 3.4.1, the dataset used for payload detection contains the Cartesian coordinates (XYZ values) of the drone in the internal coordinate frame of AirSim©. This dataset can be repurposed to evaluate the localization algorithm's performance. While the BB information can be extracted using the YOLO object detection method, AirSim© itself provides an internal feature called "object detection". This feature allows access to the BB information directly from the simulator without the need to run an object detection network on the data. The advantage of using the internal object detection feature is that it ensures consistent BB information for all recorded frames, eliminating the possibility of missed detection that can occur when using DL-based detection networks. Moreover, this feature provides GT information about the BB, which can be valuable for training object detection or classification deep networks to avoid the annotation step (although this advantage falls outside the scope of the current study). Additionally, using the internal object detection feature increases the number of frames in the dataset, as drone detection is available for all frames. However, it is important to note that this approach does not account for

the uncertainty typically associated with deep networks during the object detection stage. The uncertainty manifests itself in accurately determining the BB around the detected drone, directly impacting the performance of the presented localization methods. While the performance of the localization methods will be further analyzed using experimental data in the upcoming sections, the use of this "perfect" BB information from AirSim© object detection feature allows us to evaluate the performance of localization algorithms without the influence of external factors. This allows for an initial evaluation of the localization performance in an ideal setting.

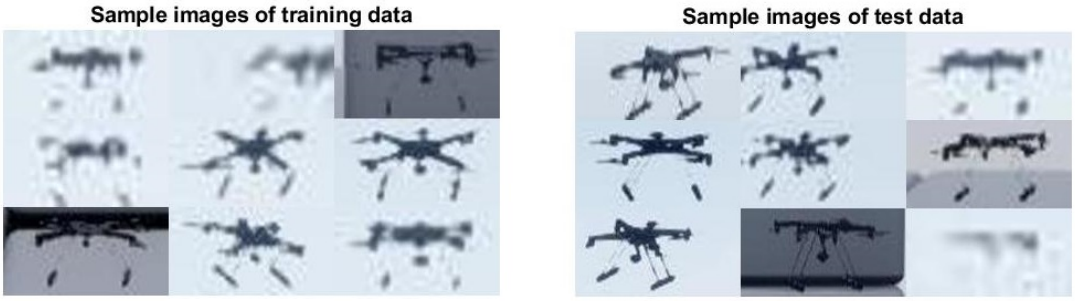
The simulated dataset will serve as a valuable benchmark for assessing the accuracy and reliability of the localization algorithms. By examining the performance of the simulated data, insights can be gained into the strengths and limitations of the methods before applying them to real-world scenarios.

The simulation section involves training the CNN network for 3D localization using various scenarios with different levels of complexity. The dataset used for training includes multiple simulation conditions, as outlined in (Table 3.1). These conditions encompass factors such as weather, payload status, environment, and drone type, enabling the exploration of diverse recorded vision data.

To train the CNN network shown in Figure 4.2, the stochastic gradient descent optimizer is employed with an initial learning rate of 1×10^{-5}. The network is trained for 10 epochs to optimize its performance. In each scenario, 80% of the dataset is utilized for training, while the remaining 20% is reserved for testing/validation.

To initiate the evaluation process of the 3D localization methods, the initial investigation focuses on a simple case involving one type of drone under various weather conditions. Specifically, a Quadrotor is selected as the drone model, and it is simulated in the Blocks environment under sunny, rainy, and snowy weather conditions. Each weather condition includes approximately 800 frames of simulated data, thus, the number of training and testing data would be 1920 and 480, respectively. Some samples of the training and testing images can be seen in Figure 4.8.

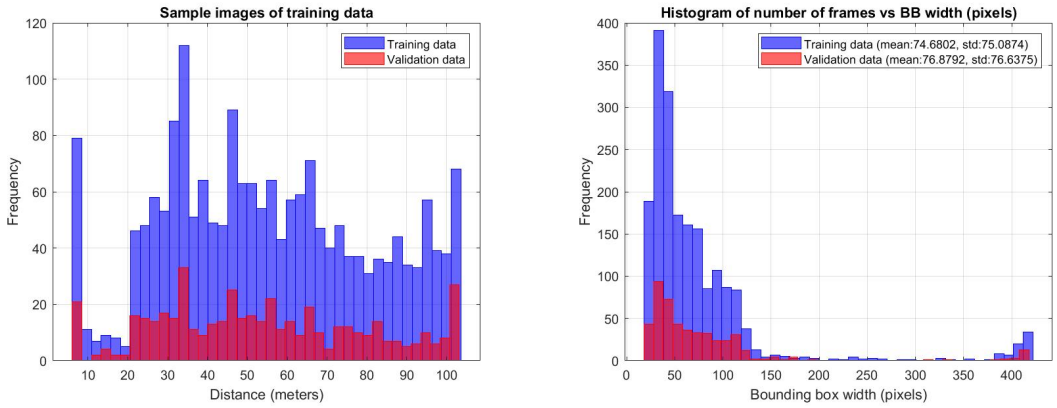
The histogram of the drone's distance and width is presented in Figure 4.9 for both the training and testing datasets. The histograms are plotted based on the size of BB before resizing to a fixed size. The analysis reveals that the majority of drone distances fall within the range of 20 to 100 meters, with a particular concentration around 30 to 40 meters. This distribution suggests that the dataset is almost well-balanced and captures various distance scenarios. Furthermore, examining the BB width distribution, it can be observed that the majority of BBs have a width below 180 pixels (with a FOV of 82 degrees and the resolution of image frames equals to 1080×1920 pixels as it was mentioned in



(a) Training data (b) Testing data

Figure 4.8: Some sample input images to the regression network for the first simulated scenario

Section 3.4.4). This width aligns with the predefined fixed size of the input images used in the CNN regression network. As a final point, the distribution of data versus distance is not uniform and as it can be seen in Figure 4.9, there is less data in some of distances. The reason is that the simulated data was captured in AirSim© with the drone being manually controlled using keyboard inputs to navigate. Consequently, the flights did not occur under conditions of "complete control," and due to human actions influencing key presses, there may be less data available for certain distances.



(a) Distance (b) BB width

Figure 4.9: Histogram of data for the first simulated scenario for localization

The scatter plot depicting the estimated distance versus the actual value is displayed in

Figure 4.10. It is evident from the plot that the estimated values deviate from the actual distances for distances less than 20 meters. This deviation can be attributed to the limited amount of available data in that range, as illustrated in Figure 4.9(a). The RMSE of the error is calculated to be 7.22 meters. However, there are two factors that offer the potential for improvement. Firstly, if we calculate the RMSE specifically for distances over 20 meters, where we have a greater amount of training data and the CNN regression network can *see* more data points, the RMSE reduces to approximately 6.4 meters. This suggests that the network performs better when it has access to a larger and more diverse dataset. Secondly, it is important to note that the training and validation data for this analysis were randomly selected from the recorded data. Consequently, continuous trajectory data, which could benefit from tracking filters like the Kalman filter or simple smoothing filters such as the MA filter, were not included. The impact of applying these filters will be discussed in the subsequent Section 4.4.3.

In summary, the scatter plot reveals discrepancies between estimated and actual distances, particularly for shorter distances with limited data availability. However, by focusing on distances over 20 meters, the performance of the CNN regression network improves, yielding a lower RMSE. Furthermore, the absence of continuous trajectory data and the application of tracking or smoothing filters are important considerations that will be explored in the forthcoming section.

The complexity of the scenario increases when we encounter both loaded and unloaded drones, as well as changing environments. In order to compare it with the previous case, we consider the loaded and unloaded Quadrotor model in both Blocks and City Park environments. Examples of the captured images can be seen in Figure 4.11.

This data exhibits more diverse backgrounds, and the distribution of backgrounds may differ between the training and test datasets. However, despite these challenges, the CNN network was able to estimate the distance, as shown in Figure 4.13. The performance of the distance estimation decreased in this more difficult scenario compared to the previous one (Figure 4.10). The RMSE for distance estimation was calculated to be approximately 9.24 meters in this scenario. The distribution of BB width remained similar to the previous scenario (Figure 4.12(b)), but the distribution of distance became denser around 40 meters, indicating that we had more data points around this value (Figure 4.12(a)). Consequently, the network achieved better results for distances around 40 meters due to the abundance of data for this particular case. The reduced deviation from the actual distance in Figure 4.13 reflects this improved performance.

The inclusion of various types of drones with different sizes and shapes adds another layer of complexity to the scenario. In the third simulated scenario, we simulate four

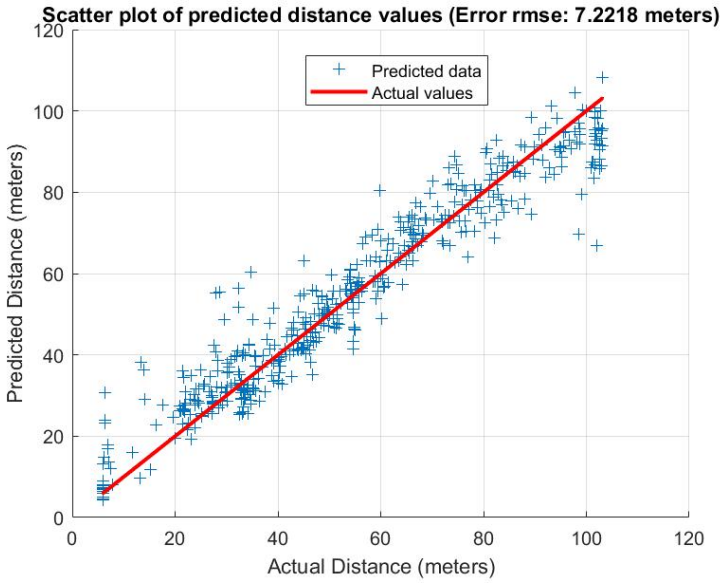
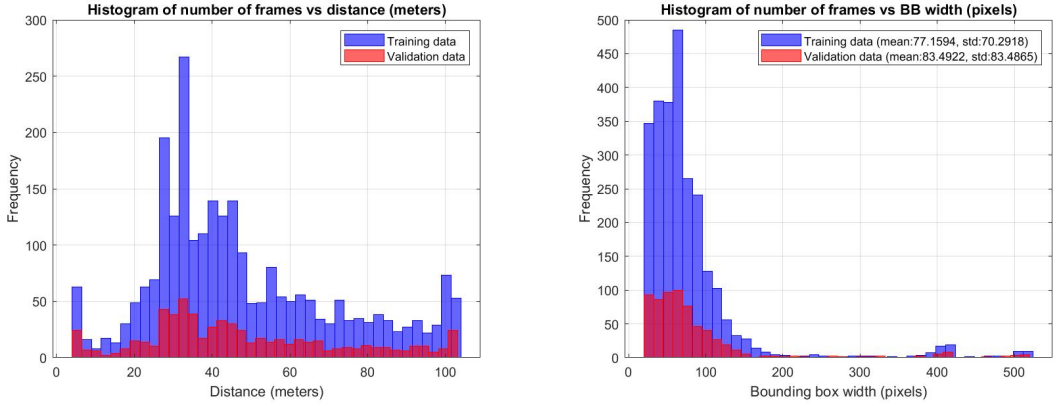


Figure 4.10: Scatter plot of the estimated distance vs. the actual value for the first simulated scenario

(a) Training data (b) Testing data

Figure 4.11: Some sample input images to the regression network for the second simulated scenario

distinct drone models: DJI Mavic, DJI FPV, DJI Inspire, and Quadrotor. Each of these drone types possesses its own unique characteristics, such as differing dimensions and shapes. By incorporating this diverse range of drones into the Blocks environment, we create a more challenging distance regression problem. Samples of input images for this scenario can be seen in Figure 4.14.



(a) Distance (b) BB width

Figure 4.12: Histogram of data for the second simulated scenario for localization

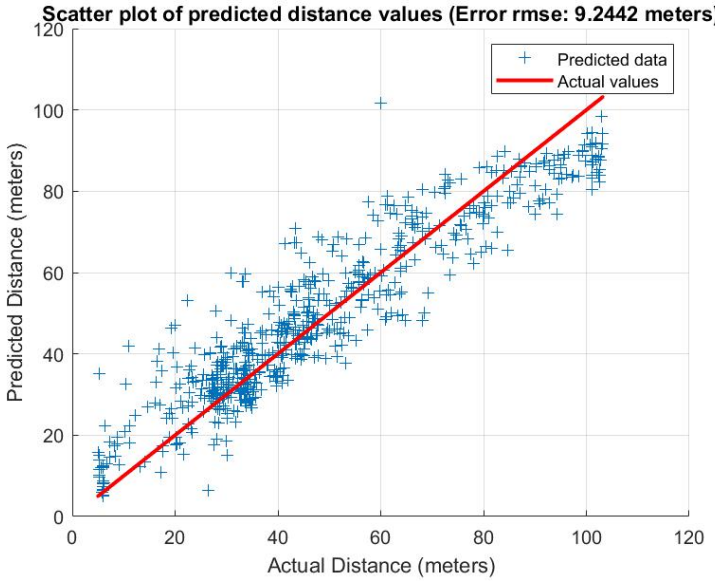
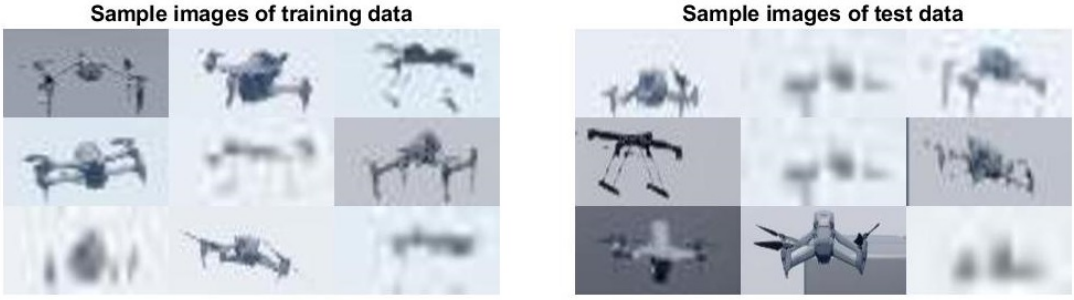


Figure 4.13: Scatter plot of the estimated distance vs. the actual value for the second simulated scenario

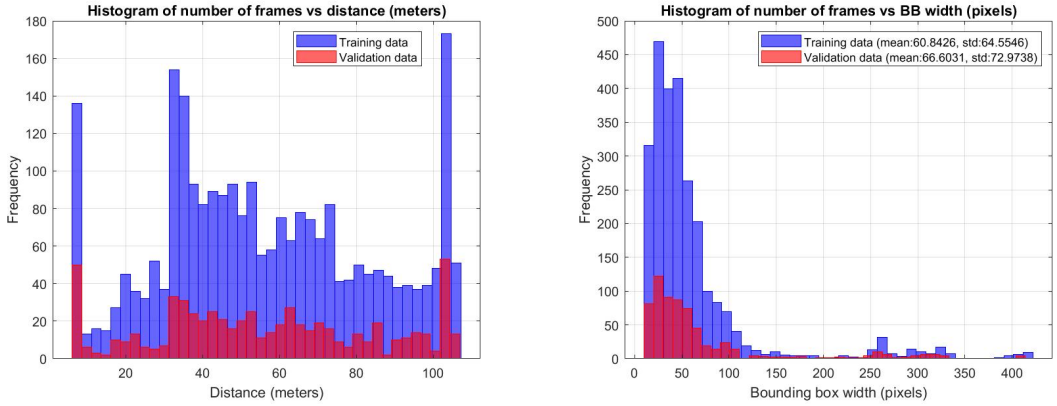
The histogram in Figure 4.15 depicts the distribution of distance values in the dataset. It shows that we still have a considerable number of observations with distances ranging from 10 meters up to approximately 100 meters. As it is expected, the error has increased in this scenario compared to the first one, where we only had one type of drone. The



(a) Training data (b) Testing data

Figure 4.14: Some sample input images to the regression network for the third simulated scenario

RMSE now increased to around 7.8 meters, indicating a higher level of uncertainty in the distance estimation for this more complex scenario.



(a) Distance (b) BB width

Figure 4.15: Histogram of data for the third simulated scenario for localization

The aforementioned results highlight the efficacy of the CNN regression network in estimating the distance of the detected drone within the bounding box. The calculated RMSE indicates that this method can achieve accuracy levels comparable to GPS measurements. However, it is important to note that the simulated dataset includes various other backgrounds and conditions. As a path for future research, this allows a comprehensive evaluation of the method's performance and opens up the possibility for its effectiveness

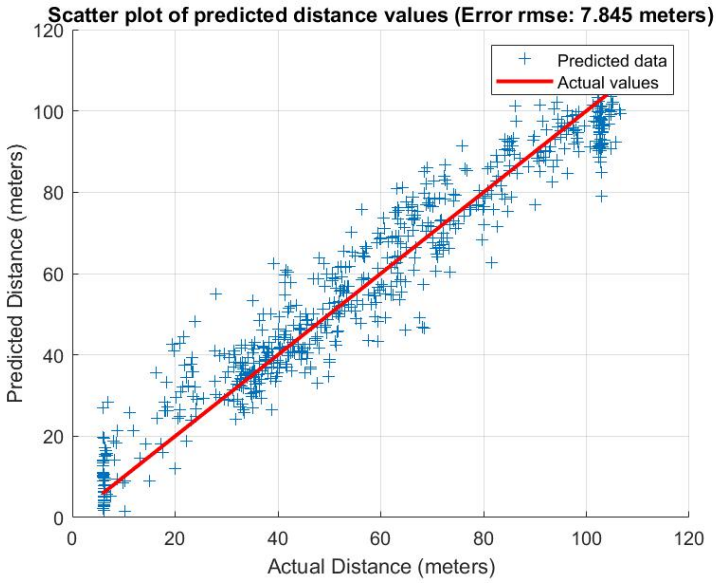


Figure 4.16: Scatter plot of the estimated distance vs. the actual value for the third simulated scenario

and robustness in more diverse scenarios. In the next sections, the performance is analyzed using real UAV in both fixed-on-the-ground and flying positions.

4.4.2 Fixed Camera and Drone on the Ground

In this section, we present the results obtained from the experimental data collected using a fixed drone on the ground. The dataset was collected at both the University of Ottawa (uOttawa) and the NRC campus, with a maximum distance of around 50 meters (because of the limitations of the available area).

To estimate the distance between the camera and the UAV, we utilized captured videos and compared the results with the GT values obtained from the designed test scenario using TSS (explained in the previous parts). The camera was fixed on the base station (Figure 4.17), while the UAV was positioned on a stand near the prism (Figure 4.18). The drone was aligned with the prism to have almost the same distance to the main station (Figure 4.19). This setup allowed us to establish accurate GT measurements for comparison. In this test, a DJI Phantom model was used as the target drone. The explained configuration allowed for capturing videos from different orientations of the UAV.

Figure 4.17: The fixed camera on the top of the TSS equipment

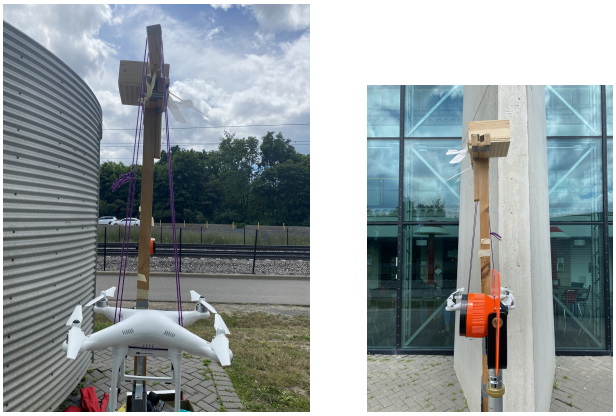
Specifically, we considered three orientation statuses: horizontal, inclined to the right, and inclined to the left (Figure 4.20). The final test configuration can be seen in Figure 4.21 and Figure 4.22. As it is clear, these configurations provide different backgrounds in the captured images.

To evaluate the performance of the previous algorithms, we estimated the distance between the camera and the UAV using the captured videos and compared it against the GT values obtained from the total station surveying. For the distance regressor, the data was divided into two subsets: one for training and the other for testing. The training data consisted of samples collected at the NRC campus, while the uOttawa dataset served as the test data. This approach allowed us to assess the algorithms' performance on unseen data and evaluate their ability to generalize to different environments.

Examples of training and testing data, which are resized images used as inputs to the regression network, are depicted in Figure 4.23 and Figure 4.24, respectively. The drone was suspended on a stand during data collection, resulting in minimal variation in the background and drone shape for each distance.

The distance estimation results are presented in Table 4.1. While the results indicate that the proposed method achieves satisfactory performance in estimating the distance,

Figure 4.18: The position of the UAV close to the prism of the TSS equipment



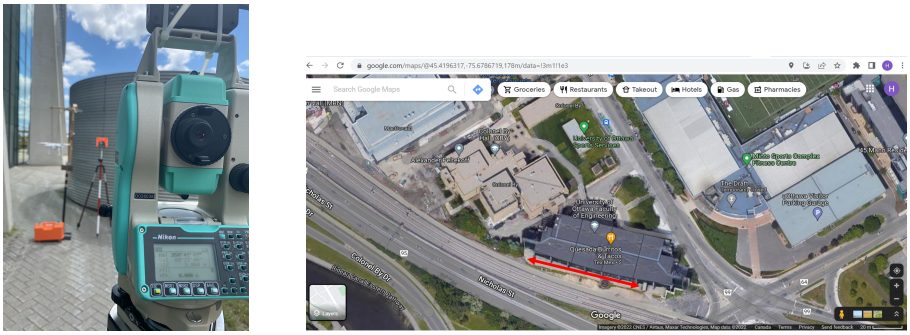
(a) (b)

Figure 4.19: The alignment of the UAV and the prism

it is important to note that the recorded video data, where the drone was suspended on a stand, lacks diversity and randomness for each distance. In addition, because of the unseen background by YOLO object detection method, it could not detect the drone in

(a) Inclined to the left (b) Horizontal (c) Inclined to the right

Figure 4.20: Various orientations of the UAV



(a) The setup (b) Aerial view of the test area

Figure 4.21: The test setup at the UOttawa campus

Figure 4.22: The test setup at the NRC campus in Montreal Rd., Ottawa, Ontario

most ground-mounted scenarios and thus, the fixed bounding box was assigned to these videos (there is not any uncertainty in the extracted region of interest). As a result of these two mentioned points, the standard deviation is relatively small. A more comprehensive analysis can be conducted using flight data recorded during the experimental test, which

Figure 4.23: Samples of the ground-mounted training dataset

Figure 4.24: Samples of the ground-mounted testing dataset

will be discussed in the next section. As a final point regarding YOLO's object detection performance, it should be mentioned that the dataset was recorded under limited scenarios, which resulted in a lack of diversity in the backgrounds. Consequently, the network did not have exposure to a wide range of cases, affecting its ability to detect drones. It is important to note that, firstly, the detection procedure is beyond the scope of this thesis. Secondly, in practical scenarios, ground-to-ground drone views are not applicable, and this setup served as an intermediary step before the final flight test.

As previously discussed in Section 4.2.1, a significant challenge in classical distance estimation arises from the orientation of the drone and its impact on the number of pixels

Table 4.1: Performance of the distance estimation method for the ground-mounted drone

Distance (meters)	$\text{Mean}_{\text{error}}$ (meters)	$\text{std}_{\text{error}}$ (meters)	$\text{RMSE}_{\text{error}}$ (meters)
11.953	1.5591	0.4324	1.7569
20.2225	2.4837	0.4643	2.8253
35.421	4.0623	0.4821	4.22
50.896	6.56	0.4882	6.7043

it occupies in the captured image. The classical approach estimates distance solely based on the width of the BB measured in pixels. However, since the BB is typically assumed to be horizontally oriented, it does not accurately represent the drone's true size when the drone is rotated. Consequently, this often leads to inaccurate distance estimation. In contrast, DL-based methods can handle oriented drone samples if they are included in the training dataset, allowing the network to learn how to handle such cases. As demonstrated in the previous section regarding simulated data results, which included various oriented drone samples (see sample images), the error in distance estimation did not significantly increase. This is in stark contrast to the classical approach, where the relationship between BB size and estimated distance is more direct. In cases involving oriented drones, it is advisable to extract oriented BB representations instead of relying on horizontal ones. One possible approach involves processing the drone image within the BB to estimate the orientation angle, which can then be used to extract the oriented BB. In the ground-based test setup, scenarios involving oriented drones are available (see Figure 4.20), making them suitable for testing. One proposed orientation estimation approach converts the RGB image of the drone into grayscale, applies a threshold to produce a black-and-white image, and identifies the largest cluster of connected white pixels. Two test cases illustrating this approach can be observed in Figures 4.25 and 4.26. These examples demonstrate how this approach can generate an oriented BB around the drone, facilitating more accurate distance estimation.

It is important to note that this image processing algorithm may not be easily applicable in practical scenarios with diverse backgrounds in the captured scene. Therefore, a more practical solution is to utilize the DL-based method to address these cases, especially considering their performance as demonstrated in the simulated data results (Section 4.4.1).

Figure 4.25: Sample image of the oriented drone and the process of extracting BB (GT and estimated distance equal to 3.944 and 4.090 meters (classical approach), respectively)

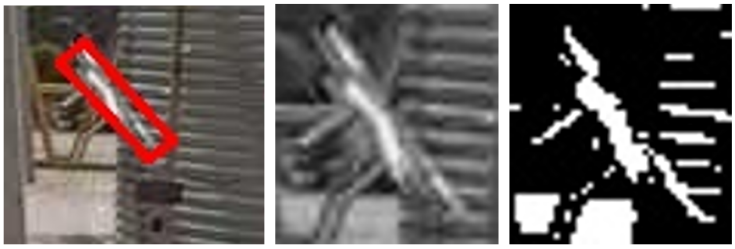


Figure 4.26: Sample image of the oriented drone and the process of extracting BB (GT and estimated distance equal to 11.953 and 12.639 meters (classical approach), respectively)

4.4.3 Experimental Flight Test

The next phase involves analyzing the performance of the localization methods using experimental data collected in an air-to-air configuration. Specifically, as detailed in Section 3.5.2, a camera was mounted on the observer drone to capture videos of the target drone positioned in front of it. The general specifications of the experimental data were outlined in that section. For the specific purpose of localization testing, it is important to note that the GT values for location information were obtained using GPS sensors installed on both drones. The GPS information was then relayed to the ground station as telemetry data. By utilizing the GPS sensor, the latitude, longitude, and altitude information of both drones were recorded over time. Using GPS to Cartesian coordinate converter tools, the XYZ coordinates of the drones relative to each other could be derived. Consequently, the spherical coordinates including the range (i.e., the distance between the camera and the target drone), azimuth, and elevation angles could be easily calculated based on this information. These values serve as the ground truth for comparison with the results obtained from the localization estimation methods.

It is worth mentioning that the calculation of these values presented a challenge due to the lack of synchronization and identical sampling frequencies in the telemetry data time vectors. Therefore, as a preprocessing step, both sets of location data (from the observer and target drones) needed to be interpolated with a common reference time vector to ensure the calculation of their relative location.

Other specifications of the recorded data align with those described in Section 3.5.2. The distance between the observer and target drones varied between 20 and 100 meters. While the observer drone was hovering, the camera recorded videos of the target drone, which performed various maneuvers, including approaching and moving away from the observer drone, moving left and right (to capture a range of GT azimuth angles relative to the camera), and ascending and descending (to encompass a range of GT elevation angles).

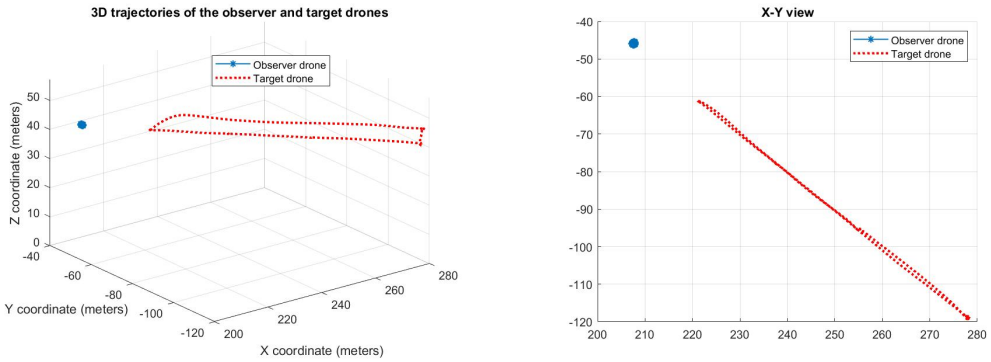
This diverse set of flight scenarios was designed to assess the performance and robustness of the localization estimation methods under various conditions and angles. By evaluating the estimated results against the GT values, we can gain valuable insights into the accuracy and effectiveness of the proposed methods in real-world scenarios. In the following, we will present the analysis and results derived from this experimental data. The set of scenarios included different combinations of the following trajectories (Figure 3.15):

- Hovering observer drone with camera and flying target drone in forward and backward directions in front of the observer drone, and hovering at some witness points with distances equal to around 20, 35, 45, 60, 70, and 100 meters from the camera.
- Hovering observer drone with camera and flying target drone in upward and downward directions for around 10 meters in front of the observer drone
- Hovering observer drone with camera and flying target drone in left and right directions for around 10 meters in front of the observer drone

In the DL-based distance estimation method, we adopt the same set of hyperparameters as discussed in Section 4.4.1, with the exception of the learning rate, which is adjusted to 1×10^{-4}. To evaluate the performance of the CNN regression network in distance estimation, we employ data extracted from one of the recorded videos during flight tests for training and validation purposes. Subsequently, the trained network is applied to the remaining recorded videos to assess its performance in different scenarios. The training dataset consists of 2269 frames containing the detected drone, which is further split into 1816 frames for training and 453 frames for validation (randomly selected). Similar to the simulation section, we utilize the RMSE of the error as a metric to analyze the performance. It is worth noting that during data collection, there were instances where the target drone

remained stationary at specific distances from the observer drone, resulting in a non-uniform distribution of distance data in some scenarios.

The 3D trajectory of both the observer and target drones in the training scenario is shown in Figure 4.27. The trajectory illustrates that the target drone flew at distances ranging from approximately 20 to 100 meters relative to the camera on the observer drone. However, there were instances where the target drone made stops at various distances, hovering in place. These hovering periods are clearly visible in Figure 4.28, which provides a histogram representation of the recorded data.

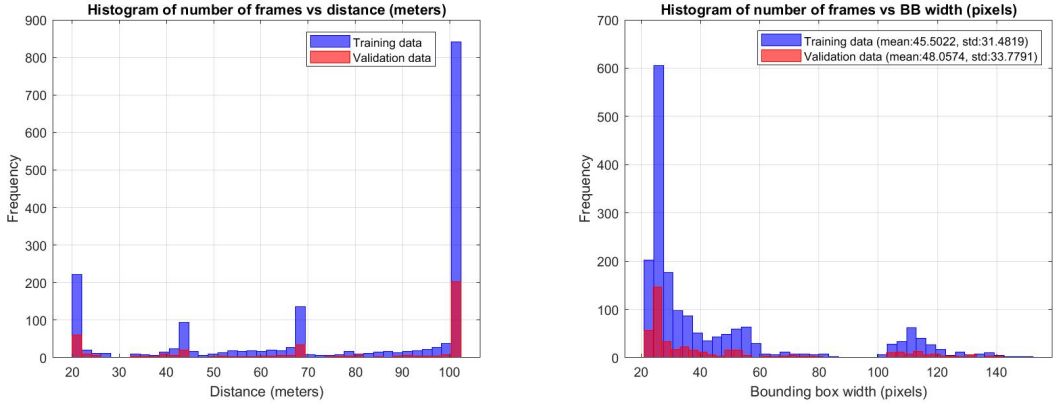


(a) 3D trajectory (b) X-Y view

Figure 4.27: Flight trajectory of the drones during the scenario used for training

The histogram of the BB width is presented in Figure 4.28(b). It is observed that the distribution of distances exhibits a higher concentration of data points around 100 meters. Consequently, one would expect the histogram of the BB width to be concentrated around smaller values. As the target drone moves farther away, its size in the captured frame reduces, resulting in a decrease in the size of the corresponding BB. This correlation between distance and BB width is clearly depicted in the histogram, emphasizing the inverse relationship between the two variables.

A selection of sample images from the training data depicting the drone can be observed in Figure 4.29. These images highlight the presence of diverse backgrounds and varying lighting conditions within the training dataset. Such diversity is essential for training DL-based methods, as it enables the model to learn robust and generalizable features that can effectively handle variations in real-world scenarios. The inclusion of different backgrounds and lighting conditions enhances the model's ability to adapt and accurately estimate the distance of the drone regardless of the environmental settings.



(a) Distance (b) BB width

Figure 4.28: Histogram of data for the training data in the experimental test for localization

Figure 4.29: Some sample input images to the regression network for the training data in the experimental test

Because of the random splitting of the data for training and validation sets, the scatter plot of estimated distance values is a better illustration of the performance of the CNN network for the validation set. On the other hand, when evaluating the network's performance on the test data, which consists of an entire recorded video, a different approach is adopted. The flight trajectory plot is considered there, and smoothing filters are applied to enhance the estimation accuracy. The resulting RMSE for the validation set is approximately 9.78 meters, as depicted in Figure 4.30.

In addition to estimating the distance, accurate 3D localization of the drone also requires estimation of the azimuth and elevation angles. For this purpose, the approximate

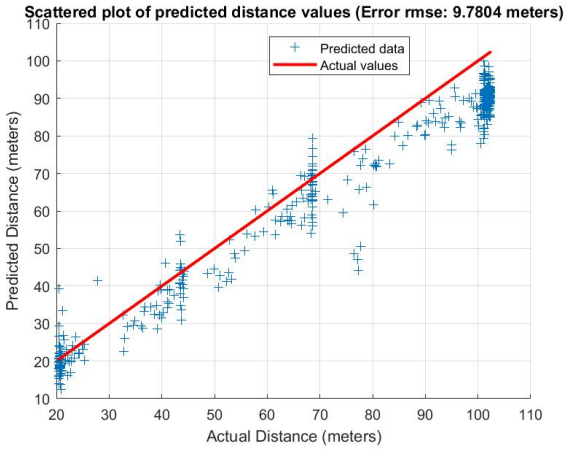
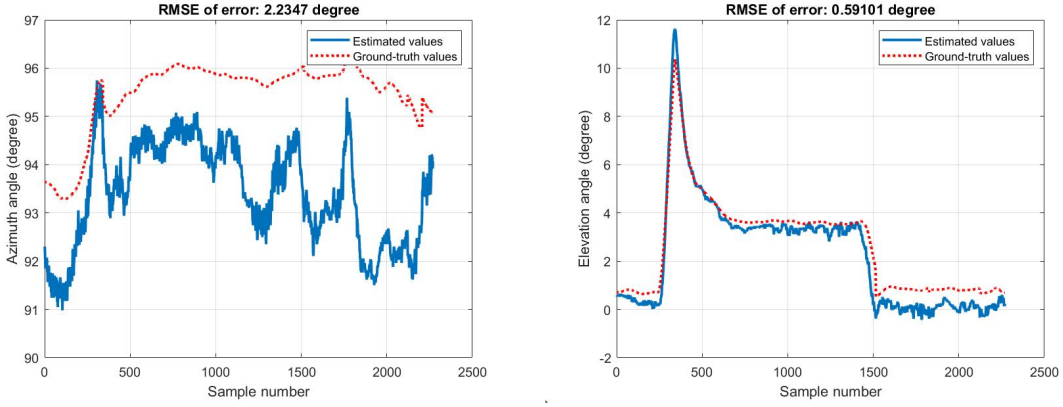


Figure 4.30: Scatter plot of the estimated distance vs. the actual value for the validation data in the experimental test

equation (4.3) can be utilized. To obtain GT values for comparison, GPS information for both the observer and target drones was converted to Cartesian coordinates. Subsequently, the XYZ coordinates of the camera and target were calculated. By employing Cartesian to spherical coordinate conversion equations, the GT values for azimuth and elevation angles were determined, which can be compared with the estimated values. Figure 4.31 illustrate the results of angle estimation for the flight scenario used during the training phase. This figure depicts the curves representing the estimated angles, which exhibit a close resemblance to the GT values. The RMSE for the azimuth and elevation angles are calculated to be 2.23 and 0.6 degrees, respectively. This indicates a relatively accurate estimation of both angles by the localization method. The small RMSE values signify the effectiveness of the utilized approach in accurately estimating the azimuth and elevation angles.

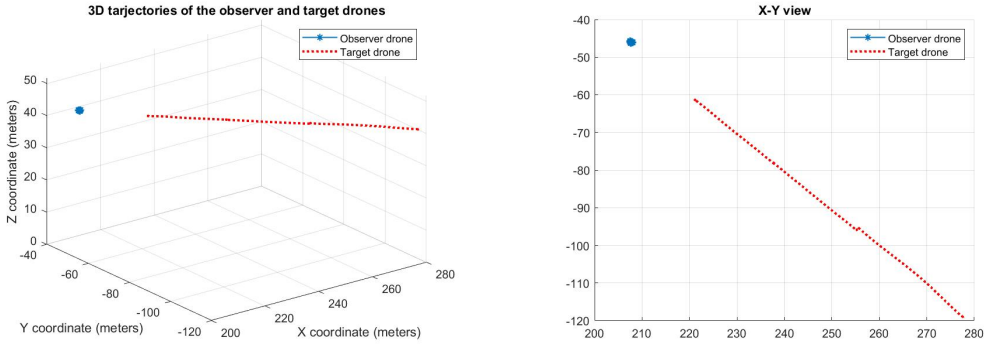
As was mentioned in the previous chapter, a series of scenarios were conducted during the experimental flight test. In this study, a subset of these scenarios has been chosen for the performance analysis of the localization method. The trajectory for the first scenario is presented in Figure 4.32. In this scenario, the distance of the target drone changed from 20 to around 100 meters.

The distribution of distances and the corresponding BB widths of the detected drone are illustrated in Figure 4.33. The data analysis shows that the target drone remained stationary at approximately 20, 45, 70, and 100 meters away from the camera for a considerable duration. Consequently, this hovering behavior of the drone leads to a distinctively centralized distribution around these specific distance values in the histogram.



(a) Azimuth angle (b) Elevation angle

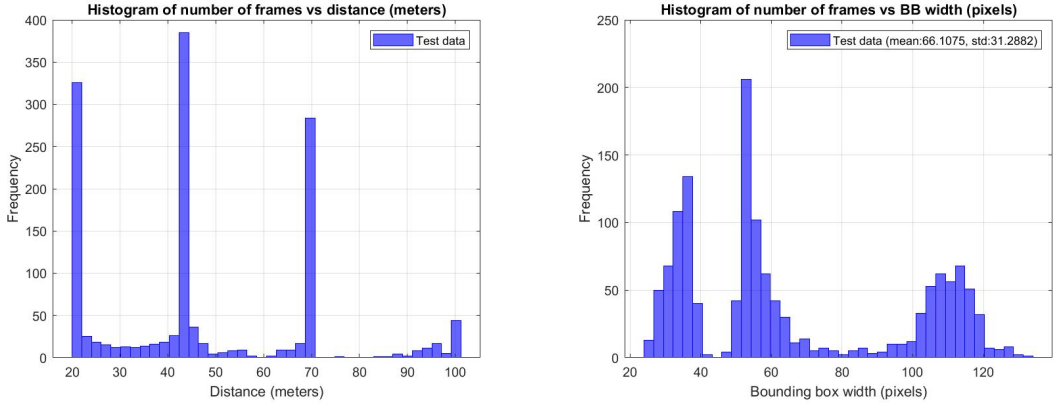
Figure 4.31: Comparison of estimated azimuth and elevation angles with the GT values from GPS information (corresponsing scenario to the training data)



(a) 3D trajectory (b) X-Y view

Figure 4.32: Flight trajectory of the drones during the first scenario in the experimental test

Samples of input images in this scenario are shown in Figure 4.34. The corresponding estimated distances are displayed in Figure 4.35. To smooth out the estimated distance curve and account for the movement of the observer drone in the general scenario, we can apply the Kalman filter to the data. While the experimental test involved the observer drone hovering in a fixed position, this is not typically the case. In most scenarios, the observer drone, equipped with a camera or sensor, will actively pursue the target drone, resulting in its own movement. Therefore, the estimated distance, azimuth, and elevation angles inherently contain information about the maneuvers of both the observer and target



(a) Distance (b) BB width

Figure 4.33: Histogram of data for the first test scenario in the experimental test for localization

drones. To apply the Kalman filter, we first need to compensate for the observer drone's maneuver. This involves transforming the measurements into a fixed-position coordinate frame. In this study, we leverage the measured distance, azimuth, and elevation angles in the camera coordinate frame, along with the GPS information providing the position of the camera (or observer drone). Using the coordinate conversion transformations explained in Section 4.3.2, we can convert the measured spherical coordinates to Cartesian coordinates in a fixed-position frame. In this test, the GPS information of the radar was known, and we assume this fixed coordinate frame to be the radar frame without loss of generality. The necessary equations for the coordinate conversion transformations are described in Section 4.3.2. Thus, after this conversion, we have the Cartesian coordinates of the target as the measurements. If we assume the state vector to include the Cartesian coordinates and the corresponding velocities, by considering the constant-velocity case, the Kalman state-space equations can be written as below:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1}$$
$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$$

In this equation, the variables can be described as follows:

- $\mathbf{x}_k$ is the state vector at time step k. It contains the following variables:

$$x_k : \text{X-coordinate of the target}$$
$$y_k : \text{Y-coordinate of the target}$$
$$z_k : \text{Z-coordinate of the target}$$
$$\dot{x}_k : \text{X-velocity of the target}$$
$$\dot{y}_k : \text{Y-velocity of the target}$$
$$\dot{z}_k : \text{Z-velocity of the target}$$

- $\mathbf{z}_k$ is the measurement vector at time step k. It contains the following variables:

$$x_k : \text{X-coordinate of the target}$$
$$y_k : \text{Y-coordinate of the target}$$
$$z_k : \text{Z-coordinate of the target}$$

- $\mathbf{A}$ is the state transition matrix. It represents the relationship between the state at the current time step and the state at the previous time step. For the constant velocity model, it is given by:

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{3\times3} & \Delta t\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}$$

  where Δt is the time step between measurements, $\mathbf{I}$ is the identity matrix, and $\mathbf{0}$ is the zero matrix. For the camera, Δt is the time difference between consecutive frames. Since the frame rate for the camera was 30 frames per second, Δt equals to 1/30 or 0.033 seconds.

- $\mathbf{H}$ is the measurement matrix. It maps the state space to the measurement space. In this case, it extracts the position information from the state vector. It is given by:

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}$$

  where $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ is the zero matrix.

- $\mathbf{w}_{k-1}$ is the process noise vector at time step $k-1$. It represents the uncertainty or disturbance in the state transition. It is assumed to be zero-mean Gaussian noise.

- $\mathbf{v}_k$ is the measurement noise vector at time step k. It represents the noise or uncer-

tainty in the measurements. It is assumed to be zero-mean Gaussian noise.

- $\mathbf{Q}$ is the covariance matrix of the process noise. It captures the variance and covariance of the process noise terms. It is given by:

$$\mathbf{Q} = \begin{bmatrix} \sigma^2_{\text{accel}}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \sigma^2_{\text{velocity}}\mathbf{I}_{3\times3} \end{bmatrix}$$

  where σ_{accel} is the standard deviation of the acceleration and σ_{velocity} is the standard deviation of the velocity measurements.

- $\mathbf{R}$ is the covariance matrix of the measurement noise. It captures the variance and covariance of the measurement noise terms. It is given by:

$$\mathbf{R} = \begin{bmatrix} \sigma^2_{\text{x}} & 0 & 0 \\ 0 & \sigma^2_{\text{y}} & 0 \\ 0 & 0 & \sigma^2_{\text{z}} \end{bmatrix}$$

  where σ_{x}, σ_{y}, and σ_{z} are the standard deviations of the X, Y, and Z coordinates, respectively. Assuming decoupled case for the Cartesian uncertainties, the covariance matrix $\mathbf{R}$ will be diagonal. Note that since the actual measurements were in the spherical coordinate system, the values of the elements of $\mathbf{R}$ should be be calculated using the inverse of equation (4.5).

These equations and matrix components form the basis of the Kalman filter for tracking a target using Cartesian measurements. Since the measurements are initially in the spherical coordinate system, the resulting Cartesian coordinates from the Kalman filter are transformed back into the spherical coordinate system to ensure comparability. This transformation allows us to represent the estimated coordinates in the same coordinate system as the original measurements, and thus they will be comparable. The black curve shows the result of applying Kalman shown in Figure 4.35. The application of this filter resulted in a reduction of the RMSE of the error to approximately 7.4 meters (around 1 meter for the RMSE of the whole curve). While the RMSE may not show a significant reduction, the Kalman filter is capable of tracking the trajectory and effectively compensating for spikes in the distance curve. Despite the limited improvement in RMSE, the filter successfully maintains a consistent estimation of the target's trajectory throughout. Finally, the curves of estimated azimuth and elevation angles compared to the GT values are provided in Figure 4.36 which show an RMSE of 2.07 and 0.52 in the azimuth and elevation angles, respectively.

Figure 4.34: Some sample input images to the regression network for the first scenario in the experimental test

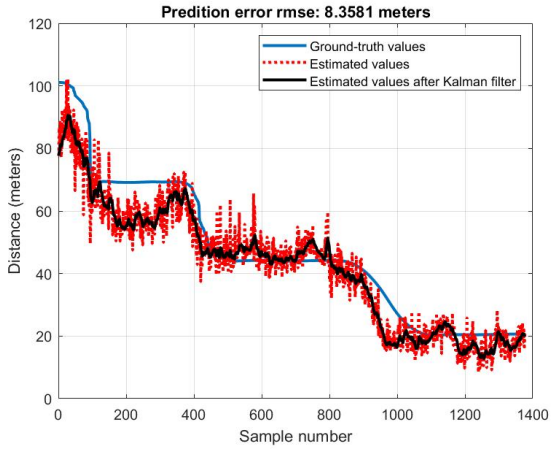
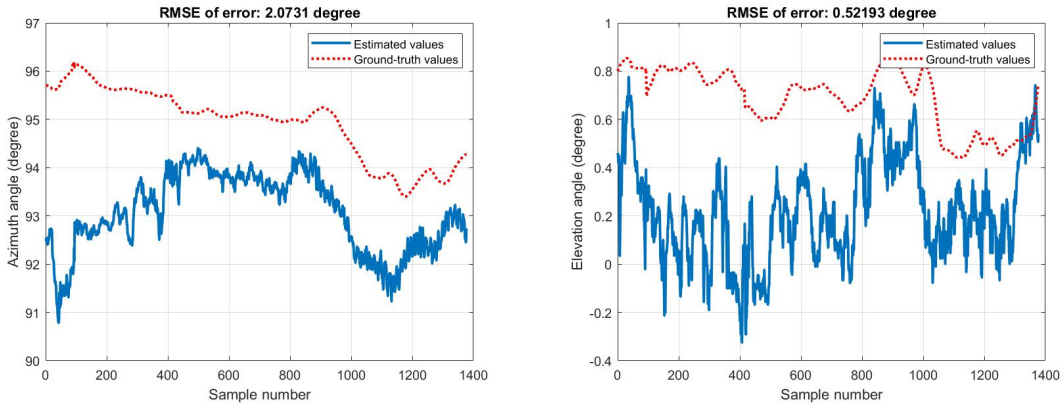


Figure 4.35: Estimated distance and the corresponding GT values for the first scenario in the experimental test

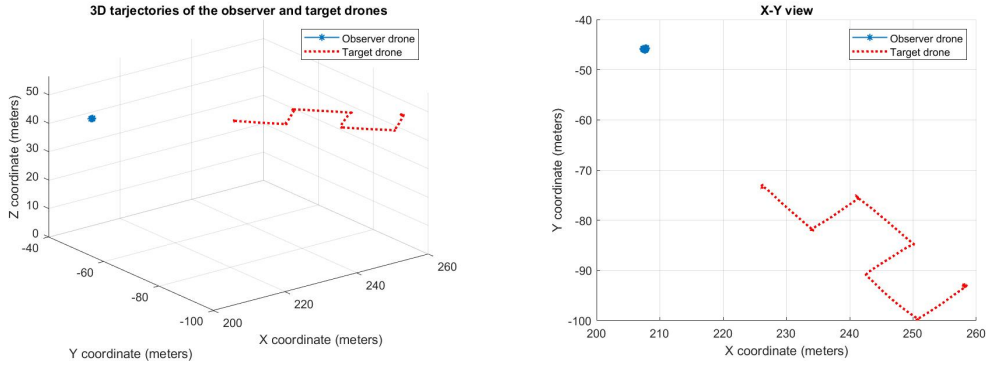
The 3D trajectory depicted in Figure 4.37 represents another scenario where the distance of the target drone ranged from 35 to 70 meters. Unlike the previous scenarios, this trajectory involved lateral movements, with the drone flying both left and right instead of following a straight path. This variation in movement patterns introduced additional view angles that were not present in the training data. Consequently, the estimation of distance in this scenario offers an opportunity to assess the network's performance in handling different orientations and angles of the target drone.

The histogram shown in Figure 4.38 reveals a distinct concentration of data points at



(a) Azimuth angle (b) Elevation angle

Figure 4.36: Comparison of estimated azimuth and elevation angles with the GT values from GPS information (first test scenario)

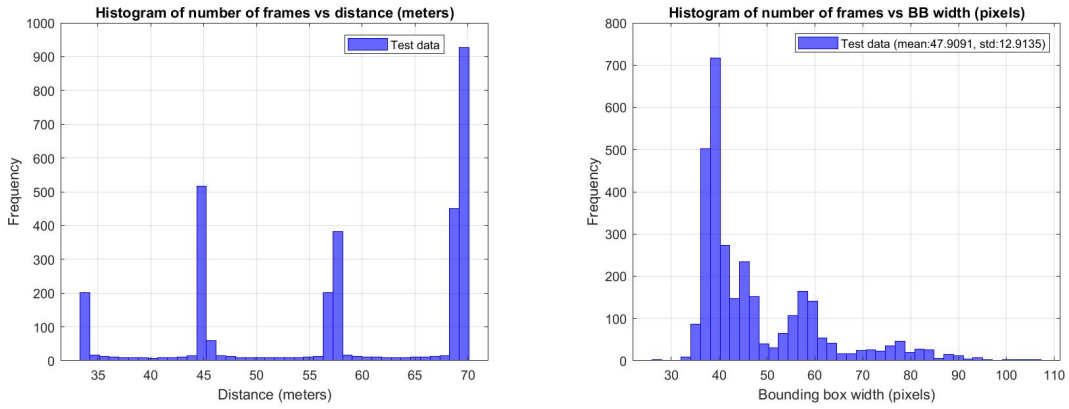


(a) 3D trajectory (b) X-Y view

Figure 4.37: Flight trajectory of the drones during the second scenario in the experimental test

distances approximately equal to 35, 45, 60, and 70 meters. This concentration indicates that the target drone remained stationary at these specific distances for a significant period of time during the recorded video.

A collection of sample images depicting the drone in the second scenario is presented in Figure 4.39. This figure highlights the existence of diverse view angles captured during this specific scenario. The variation in perspectives adds an additional challenge to accurately estimating the drone's distance. Furthermore, by evaluating the performance of



(a) Distance (b) BB width

Figure 4.38: Histogram of data for the second test scenario in the experimental test for localization

the distance estimation algorithm, it is observed that the RMSE of the estimation error is approximately 5.3 meters (Figure 4.40). However, employing the Kalman filter reduces the RMSE to 5 meters.

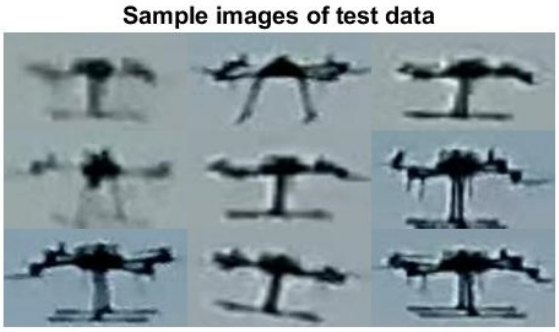


Figure 4.39: Some sample input images to the regression network for the second scenario in the experimental test

As mentioned previously, based on the trajectory shown in Figure 4.37, the drone performed lateral movements during this scenario, flying left and right. Consequently, it is expected that the azimuth angle of the target drone would exhibit distinct steps or changes corresponding to these lateral movements. These changes in the azimuth angle

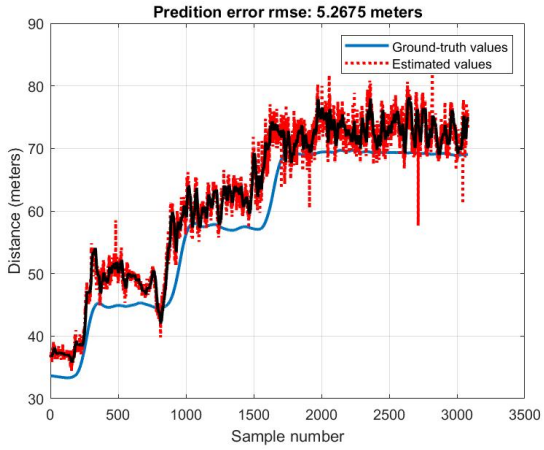
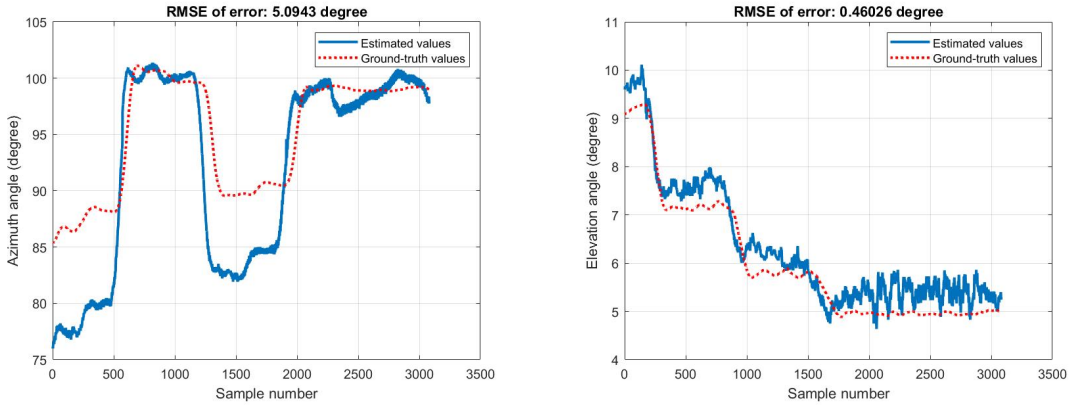


Figure 4.40: Estimated distance and the corresponding GT values for the second scenario in the experimental test

can be observed in Figure 4.41, where two significant steps of approximately 15 degrees each can be identified during the course of this scenario.

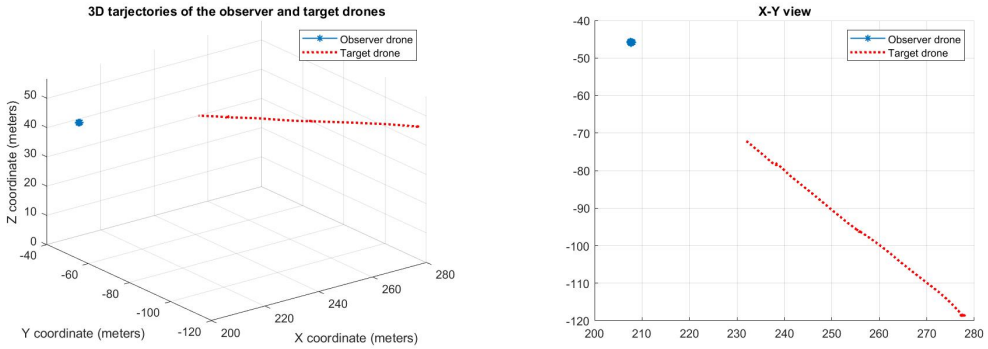


(a) Azimuth angle (b) Elevation angle

Figure 4.41: Comparison of estimated azimuth and elevation angles with the GT values from GPS information (second test scenario)

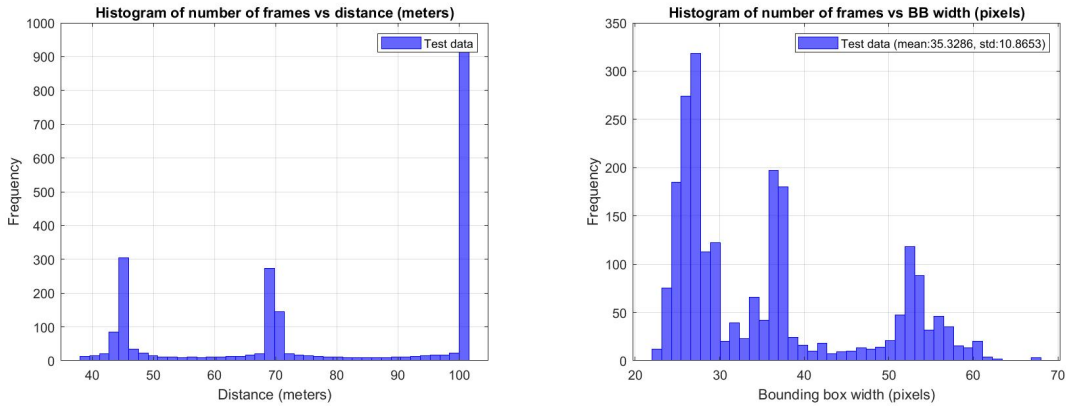
As another sample scenario, the trajectory for the third scenario is presented in Figure 4.42. In this scenario, the distance of the target drone changed from 45 to around 100 meters. The distribution of distances and the corresponding BB widths of the detected

drone are illustrated in Figure 4.43.



(a) 3D trajectory (b) X-Y view

Figure 4.42: Flight trajectory of the drones during the third scenario in the experimental test



(a) Distance (b) BB width

Figure 4.43: Histogram of data for the third test scenario in the experimental test for localization

Samples of input images in this scenario are shown in Figure 4.44. The corresponding estimated distances are displayed in Figure 4.45. It is important to mention that the limited difference observed between the RMSE of the filtered and raw data (8 and 8.6 meters, respectively) can be attributed to the prominent bias present in distances around 100 meters (Figure 4.43(a)). Another noteworthy aspect of this figure is the deviation

observed in the estimated distance around sample 1000. This deviation can be attributed to the captured images during a specific period when the drone was rotating in place. Since the training data provided to the network did not include various view angles of the drone, the rotational movement caused an inconsistency in the estimated distance. To illustrate this, a sequence of selected frames capturing the drone's rotation is presented in Figure 4.46, clearly demonstrating the turning motion. This highlights the need for training data that encompasses a wider range of drone orientations to improve the accuracy of distance estimation in such scenarios.

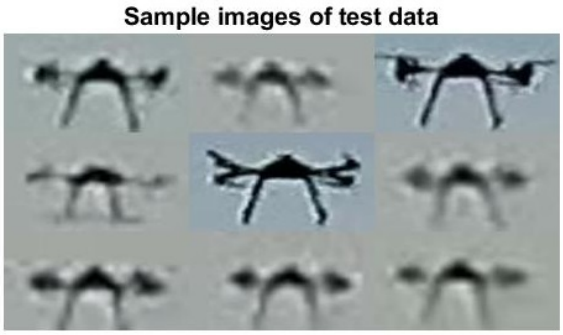


Figure 4.44: Some sample input images to the regression network for the third scenario in the experimental test

Finally, the results of angle estimation are presented in Figure 4.47. Similar to the previous curves, the simple angle estimation method demonstrates the ability to accurately track changes in the elevation angle, as indicated by an RMSE of less than 1 degree across all scenarios. This signifies the effectiveness of the method in estimating the drone's vertical position with high accuracy. However, when it comes to the azimuth angle, although the estimated curve closely follows the general trend of the ground truth values, there is a slightly higher RMSE compared to the elevation case. This suggests that the angle estimation for the drone's horizontal position may exhibit slightly more deviation from the ground truth.

Similar to the scenario of the moving camera in the payload detection problem discussed in the previous chapter, localization can also be evaluated under similar conditions. In the flight scenario presented in Figure 3.21, the estimated distance and the corresponding ground truth (GT) values, based on GPS information from both the target and observer drones, are displayed in Figure 4.48. This figure clearly demonstrates that distance estimation remains feasible even when dealing with a moving camera setup.

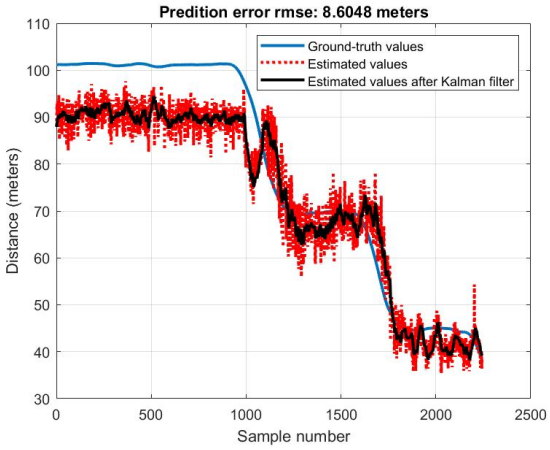


Figure 4.45: Estimated distance and the corresponding GT values for the third scenario in the experimental test

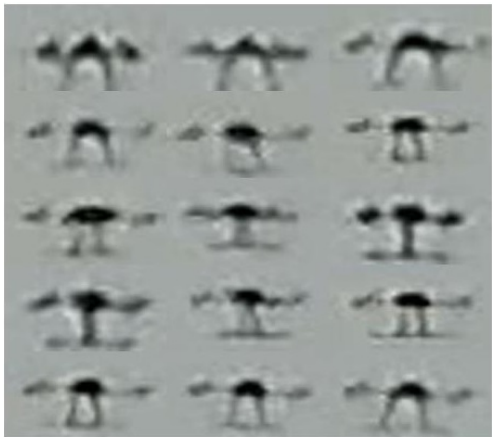
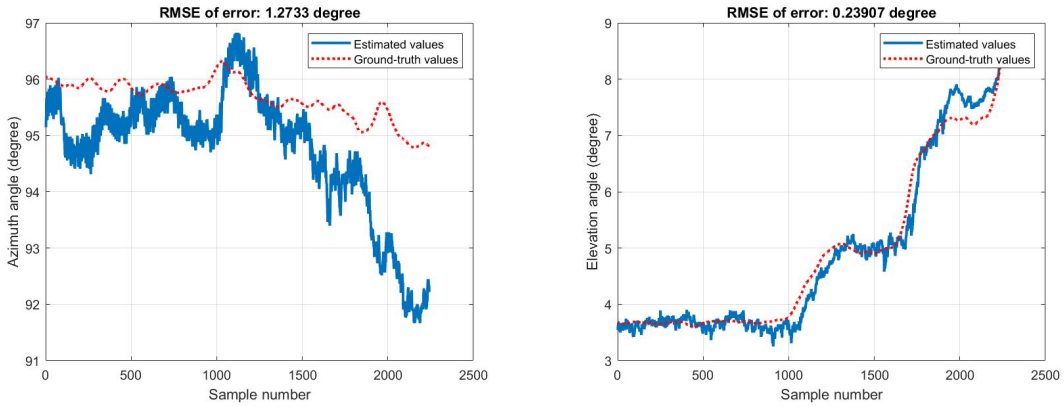


Figure 4.46: Sequence of selected frames (following a left-to-right order in each row) in the period of turning around for the third scenario in the experimental test

In all the aforementioned scenarios, the RMSE of the error using the whole trajectory data was employed as the performance metric. Although this metric provides insight into the capability of the method in the estimation of the distance, it does not reveal the error trend across different distances, alone. To analyze an error metric such as RMSE versus distance using simulation results, a sufficient amount of data is required at each distance point for statistical analysis through the Monte-Carlo approach. In the current case, various scenarios were conducted with different trajectories instead of repeating a single scenario



(a) Azimuth angle (b) Elevation angle

Figure 4.47: Comparison of estimated azimuth and elevation angles with the GT values from GPS information (third test scenario)

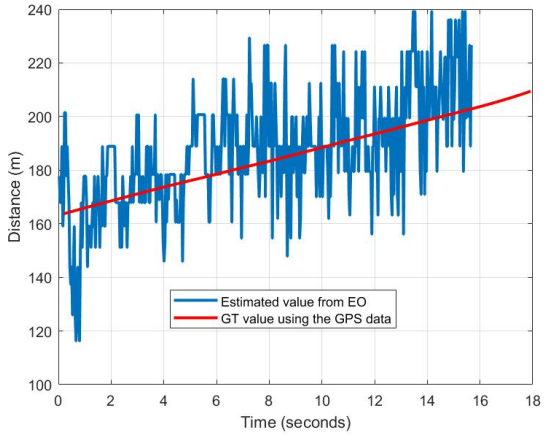


Figure 4.48: Estimated distance and the corresponding GT values for the scenario with flying target and observer drones

multiple times, which is necessary for the desired form of analysis. Consequently, we do not possess multiple data points for each distance value. Nonetheless, during the field test, the drone hovered (remained) at discrete distances. Thus, we can perform the analysis on these discrete points, resulting in metrics versus distance for specific distance values rather than a continuous curve. It is important to note that the DL-based method may not consistently yield a performance curve that decreases or increases with distance. This

is due to the not-uniformly distributed training data in terms of distance, which can affect the output of the trained network. For instance, if there are more data points at farther distances compared to closer distances, the estimation error at the far point may be smaller than that at the closer point. Conversely, a closer range provides a larger drone image in the frame, allowing for the extraction of more details (features) that have the potential to enhance performance.

Upon examining the GT distance curves in the aforementioned scenarios, it becomes evident that each scenario includes three distinct distances at which the drone remained in a hovering state for a significant duration, thus yielding valuable data for analysis. The corresponding findings for the average, standard deviation, and RMSE of the errors are presented in Table 4.2 to Table 4.4. It is noteworthy that while the first scenario exhibits an increasing trend in RMSE with greater distances, the results from the other scenarios do not demonstrate a consistent upward or downward trend in relation to distance. These findings highlight the variability in error patterns across different scenarios, emphasizing the significance of a comprehensive evaluation and a uniformly distributed dataset to discern the most reliable performance metrics for drone distance estimation.

Table 4.2: Estimation performance metrics versus the actual distance (first test scenario)

Distance (meters)	$\text{Mean}_{\text{error}}$	$\text{std}_{\text{error}}$	$\text{RMSE}_{\text{error}}$
20.5	0.51	3.26	3.30
44.1	1.83	5.39	5.69
69.5	6.56	6.76	9.42

Table 4.3: Estimation performance metrics versus the actual distance (second test scenario)

Distance (meters)	$\text{Mean}_{\text{error}}$	$\text{std}_{\text{error}}$	$\text{RMSE}_{\text{error}}$
44.8	-1.52	11.8	11.9
57.4	1.72	8.94	9.11
69.3	10.95	11.02	15.54

Table 4.4: Estimation performance metrics versus the actual distance (third test scenario)

Distance (meters)	$\text{Mean}_{\text{error}}$	$\text{std}_{\text{error}}$	$\text{RMSE}_{\text{error}}$
44.7	-5.12	3.77	6.36
69.6	-10.81	3.27	11.3
101	-5.29	3.6	6.4

4.5 Developed sensor fusion simulators

In this section, we will discuss the two simulators developed for addressing counter-UAV sensor fusion problems, which involve both radar and vision camera data. In the first simulator, the initial step involves simulating vision data within the AirSim© environment. Subsequently, based on the location information extracted from AirSim©, the radar simulation is carried out in MATLAB. Conversely, in the second architecture, a GUI is employed to facilitate the preparation of necessary code for both vision (in AirSim©) and radar simulation, which is executed in Python. In the following paragraphs, we will delve into the structure of both simulators.

4.5.1 AirSim©-MATLAB-based simulated dataset

In order to address the localization problem using multiple sensors, a joint simulator combining MATLAB and AirSim© was developed as part of this thesis. The simulator encompasses various scenarios, each designed to explore different aspects of the localization process [72, 73]. In all of these scenarios, the simulator incorporates a radar system along with a PTZ camera positioned in close proximity to the radar on the ground. The distance between these two sensors is initially set to 4 meters, but it can be easily adjusted within the MATLAB environment.

Furthermore, the simulator includes the simulation of multiple observer drones equipped with cameras, as well as target drones within the AirSim© environment. These drones operate simultaneously, enabling the recording of videos from their respective perspectives. This setup allows for the collection of synchronized video data from multiple viewpoints.

The integration of both the radar and camera sensors, along with the presence of observer and target drones, offers a comprehensive platform for investigating the localization problem. With this joint Matlab-AirSim© simulator, researchers and practitioners can conduct comprehensive evaluations of multi-sensor localization algorithms. The ability

to simulate various scenarios with different sensor configurations and environmental conditions enables the assessment and comparison of algorithm performance under diverse circumstances. The recorded videos provide visual information, while the radar system contributes to the localization process by providing range and angle measurements. In addition, all the GT values for the Cartesian coordinates of the drones are available in the prepared dataset.

To develop a multi-sensor fusion dataset using MATLAB and AirSim©, the following steps were involved:

- Running a scenario in AirSim©: The simulation begins by setting up a scenario in AirSim©, which involves the presence of observer and target drones. AirSim© is a powerful simulator that provides a realistic virtual environment for drone simulations.
- Recording video and data: During the scenario, the cameras on the observer drones capture video footage of the simulated environment. These videos are recorded with a resolution of 1080×1920 pixels and saved for further analysis. Additionally, the XYZ coordinates of both the observer and target drones are recorded along with timestamps. Furthermore, the bounding box information, which provides spatial information about the detected objects, is also saved using the "object detection" feature in AirSim©.
- Asynchronous recorded data: Two or more cameras may be used in the simulation, which record data asynchronously (one models the PTZ camera on the ground and the others model the on-board cameras mounted on the observer drones). This means that the recorded data from the various cameras may not be perfectly synchronized in terms of timestamps.
- Interpolation: Since the recorded data may have variations in the timing and sampling rate, interpolation techniques are employed to estimate missing or intermediate values for the parameters obtained in the previous step. This interpolation is crucial to ensure consistency and smoothness in the estimated parameters.
- Importing data into MATLAB: The recorded XYZ coordinates of all the observer and target drones were imported into MATLAB for further processing and analysis.
- Simulating the radar: MATLAB provides a versatile environment for simulating radar systems. Using the Matlab radar toolbox, the imported XYZ coordinates of the drones are used to simulate the radar measurements and responses. This toolbox provides simulated data with the same features as a real radar. The user can easily set

various parameters of radar including the carrier frequency, antenna specifications, resolution, false alarm rate, and so on. The radar model simulates three types of output based on the provided settings. Detections, measurements, and tracks are three output formats that can be extracted from the radar simulator. The "detections" file includes the raw detection after the signal processing unit of the radar. By applying clustering to merge the detected points of each single target, the "measurements" file is created. Additionally, similar to the practical case, the tracking filters are applied to the clustered data to smooth the data, and also predict the trajectory of the target in the case of missed detection. Unfortunately, the MATLAB documentation does not explicitly specify the type of tracking filter. Consequently, both the filter type and, more importantly, its parameters cannot be accessed for adjustment within the simulator. Additionally, it should be mentioned that the radar simulation is conducted at the data level, rather than the signal level. In other words, this simulation does not provide the raw reflected signal from the environment (and consequently, topics such as beamforming); instead, it models the data that results after signal processing and the final detection algorithm. It is important to note that the radar model in MATLAB can provide estimated target range, a feature specific to pulsed or FMCW (Frequency Modulated Continuous Wave) radars, as opposed to simple continuous-wave (CW) radars, which cannot measure distance.

- Coordinate conversion: Since both radar measurements and camera-recorded data are available for the scenario, it is necessary to perform coordinate conversion for further data fusion. This involves converting the XYZ coordinates of the targets in the camera frame and aligning them with the camera-recorded data. Based on the developed mathematical equations in Section 4.3.2, all the measurements of the cameras are transformed to the radar frame.

In this thesis, various scenarios have been executed and recorded for preparing the sensor fusion dataset. The dataset, including recorded videos, radar-extracted data, and GT values for XYZ coordinates of all the drones in each scenario, is publicly available on https://github.com/CARG-uOttawa/Multiview-Air-to-Air-simulated-drone-dataset for further analysis and research purposes.

The scenarios considered in this study encompass a range of configurations, as outlined below:

- Scenario 1: This scenario involves one radar, one PTZ camera, two hovering observer drones, and two flying target drones. The observer drones remain stationary while capturing video data, while the target drones navigate through the airspace.

- Scenario 2: In this scenario, there is one radar, one PTZ camera, one chasing observer drone that actively follows a target drone, and one flying target drone. The observer drone dynamically adjusts its position to track the movement of the target drone, capturing video footage in real time.
- Scenario 3: Similar to Scenario 2, this scenario consists of one radar, one PTZ camera, two chasing observer drones, and two flying target drones. Each observer drone independently pursues a respective target drone, resulting in simultaneous video recordings from multiple perspectives.
- Scenario 4: This scenario is an extension of Scenario 3, but with an increased number of drones. It includes one radar, one PTZ camera, five chasing observer drones, and five flying target drones. Each observer drone is assigned to track and record the movements of a specific target drone, leading to a rich dataset with multiple concurrent video streams.

More than 3000 frames are recorded in each scenario. To provide a glimpse into the developed simulator, two screenshot samples captured during its execution in the MATLAB environment are shown in Figure 4.49 and Figure 4.50. These screenshots display the simulator's interface, illustrating the trajectories of the radar and the drones as they navigate the simulated environment.

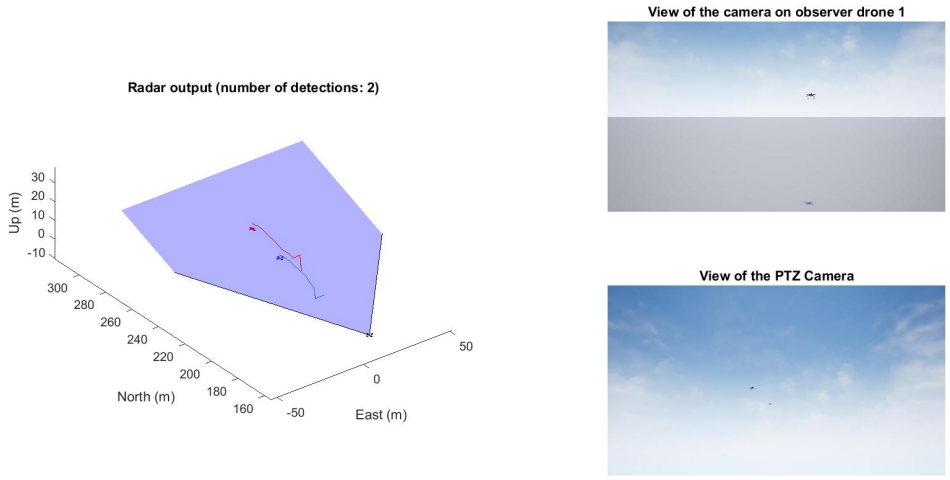


Figure 4.49: Sample screenshot of the sensor fusion simulator (scenario 2)

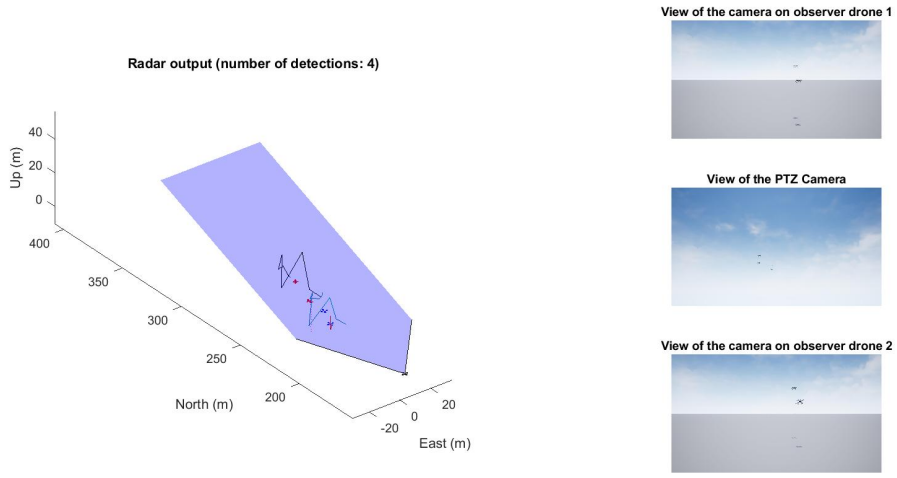


Figure 4.50: Sample screenshot of the sensor fusion simulator (scenario 3)

4.5.2 AirSim©-Python-based simulator

While the AirSim©-MATLAB-based simulator and its corresponding dataset presented in the previous section offer valuable data for research in the field of sensor fusion problems, they do come with certain limitations. Firstly, in all scenarios, drone control is achieved through keyboard inputs, which imposes constraints on available flight scenarios due to limitations in simultaneously using multiple keys. Furthermore, when using MATLAB to simulate radar, there is limited control over certain radar features, such as clustering algorithm on the detection outputs, as well as the tracking filter and its associated parameters. Lastly, with the aforementioned simulator architecture, it is not possible to achieve real-time simulation with both vision and radar data.

In response to these limitations, a new form of simulator is introduced in this thesis. The simulator's fundamental structure is built upon a Python-based GUI, empowering users to define the desired flight path for each drone interactively. Users can either draw the flight path manually or import scenario-specific data. Moreover, the GUI enables users to customize various aspects of each flight path, such as selecting the drone model and specifying the drone's altitude during flight. Additionally, within this GUI, users have the option to activate the simulator's radar system. They can configure the radar's parameters, including coverage angles in both azimuth and elevation directions, as well as its position. This user-driven flexibility enhances the versatility and utility of the simulator.

Based on the aforementioned simulator structure, the following features become accessible:

- Definition of drone models and payload shapes for individual drones.
- Simulation of various flight scenarios, both for single drones and drone swarms.
- Collection of synchronized data from various sensors with references for localization and payload detection.
- Collection of data from radar and camera(s) for sensor fusion in localization problems.
- Recording of multi-view data to enhance localization and payload detection.
- Potential integration with ML algorithms, facilitated by the Python-based framework.

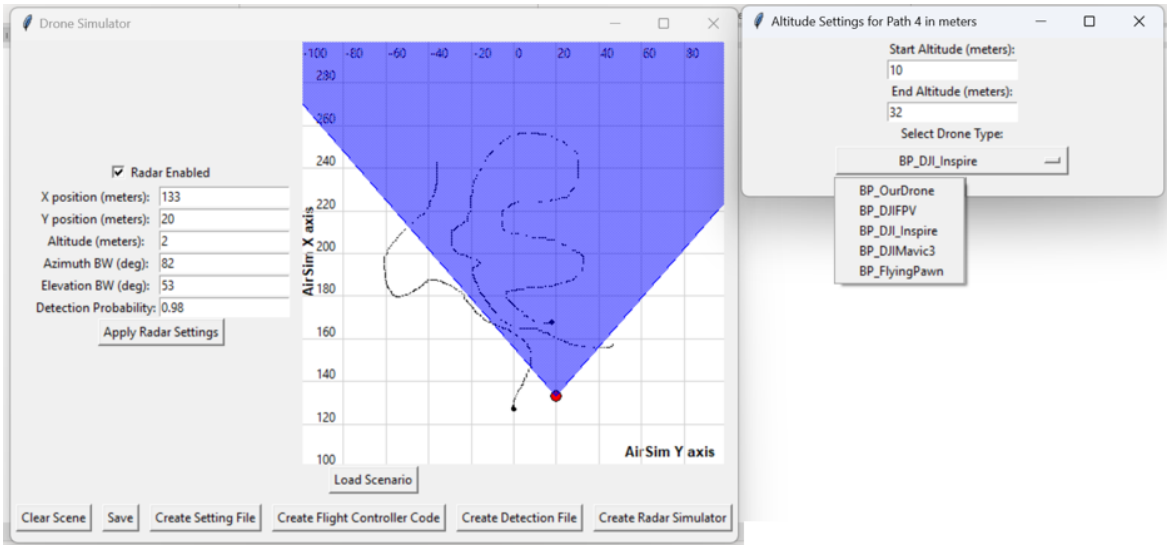


Figure 4.51: GUI of the developed simulator

The specifics of how the simulator operates and its usage instructions are outlined in appendix A. In this section, we will provide an overview of the simulator's key features. A visual representation of the GUI can be found in Figure 4.51. To set up a scenario, the user interacts with the GUI to define various parameters, including drawing the flight path (detailed instructions are available in appendix A). After generating the necessary code

using the buttons located at the bottom of the GUI, the AirSim© environment should be initiated. Subsequently, the scenario can be executed using the code generated by the GUI, enabling the recording of data. This recorded data encompasses captured images from each camera, bounding box (BB) information for each frame, and the XYZ coordinates of all cameras and target drones. There are several important points to note about this simulator:

1. When developing this simulator, it is assumed that various types of drones, both loaded and unloaded, have been imported into AirSim© using the procedure detailed in Section 3.4. Only after importing these models, the users can select the drone type for each scenario from the available list.

2. In the current version of the simulator, the radar's output is simulated at the data level, rather than at the signal level. This means that the output of the detection algorithm is assumed to be simulated within the simulator. Extending the radar model to simulate the reflected signal from the environment can be considered for future work. Additionally, raw detection outputs are simulated, but the chosen structure of the simulator allows for the application of various clustering and tracking filters, each with tunable parameters, to process this raw output.

3. In its current form, the simulator defines flight scenarios without prior knowledge of the environment. Consequently, it is assumed that there are no obstacles along the drones' flight paths. This limitation implies that all scenarios are conducted in the Blocks environment, where, beyond a certain point, only flat terrain and sky are present. For future work, expanding the simulator to handle reading real-time data from the environment and dynamically adjusting paths based on obstacle positions could be considered. Implementing such a feature would allow scenarios to be run in other environments within AirSim©.

4.6 Summary

In this chapter, we tackled the problem of extracting the 3D location of the target drone using vision data. The approach involved estimating the distance, azimuth, and elevation angles of the target drone relative to the camera. We employed a DL-based method, utilizing CNN networks as regression models for distance estimation. The azimuth and elevation angles were estimated using a straightforward pinhole model. The performance of our approach was evaluated using three types of data: simulated data, recorded data from

ground tests, and data collected during experimental flight tests. The results demonstrated the effectiveness of the proposed approach in accurately estimating 3D location information.

Additionally, to address the challenge of multi-sensor localization, an approach was developed for aligning the location information obtained from different sensors, including radar and cameras. Moreover, two simulators were developed that generate synchronized vision-radar data in a counter-UAV scenario involving one or more observer and target drones.

Chapter 5

Conclusion and future works

This chapter presents a comprehensive summary of the thesis, highlighting the key findings and contributions made throughout the study. It also outlines potential topics for future research, suggesting areas where further improvements and advancements can be explored.

5.1 Summary and contributions

Drones have gained significant popularity across diverse domains. The accurate localization and efficient detection of payloads carried by drones are essential for maintaining safety and security. This thesis presents a vision-based solution that leverages DL techniques to address these critical challenges.

Firstly, we tackled the problem of drone payload detection and classification, specifically distinguishing between loaded and unloaded drones. To address this challenge, three variants of the ResNet architecture were employed including ResNet-34, ResNet-50, and ResNet-101. These deep residual networks demonstrated excellent performance in accurately classifying the payload status of drones, achieving classification accuracy exceeding 85% in both simulated and experimental tests. The input images for these networks were obtained from the drone detector layer (the cropped image within the extracted BB for the detected drone). Additionally, given the limited availability of public datasets for drone payload detection, we took the initiative to create the first simulated dataset tailored specifically for this purpose. This dataset was developed within the AirSim© environment and comprised various drone types, simulated payloads, and diverse backgrounds.

This comprehensive dataset enabled us to train and evaluate our classification models effectively. Furthermore, to enhance the classification accuracy, we retrained the network using a combination of simulated and real-world data collected from different scenarios. This integration of simulated and practical data resulted in a significant improvement in accuracy, particularly for the experimentally collected data.

For estimating the distance of the detected drone, two distinct approaches were proposed including the classical and the DL-based algorithms. The classical approach utilizes geometric principles and computer vision techniques to estimate the drone's distance based on the size of the bounding box in the captured image. A DL-based approach was also proposed which employed a CNN regression network to estimate the distance based on visual features captured in cropped images. This approach can capture complex non-linear mappings between image features and distance, and it offers real-time distance predictions. The simple architecture of the proposed regression network help in real-time implementation. In both cases, the azimuth and elevation angles of the detected drone were estimated based on the camera's field of view and resolution, and by using the camera pinhole model. The results showed that both methods can estimate the distance of the drone with an accuracy level of GPS (less than 10 meters), as was expected from the designed methods (for the range of distances up to 100 meters). In order to reduce the deviations in the estimated values, some filters were applied to the estimated data including Kalman and MA filters.

Apart from the primary issues addressed in this thesis, we also tackled the challenges associated with the multi-sensor counter-UAV problem. To facilitate sensor fusion, particularly for localization purposes, we developed an algorithm that prepares the data by aligning measurements from both the radar and the camera mounted on the flying drone. Furthermore, we created a multi-sensor simulator that incorporates both radar and cameras. This simulator serves as a valuable tool for conducting further research and experimentation in the realm of counter-UAV measures.

5.2 Future works

Regarding payload detection, although Chapter 3 has provided a detailed analysis of the payload detection problem using the ResNet network architecture on both simulated and experimental datasets, there are several areas that can be explored in future work to further improve the performance and applicability of the proposed approach.

Firstly, the current study focused on a binary classification problem of distinguishing between loaded and unloaded drones. However, there is potential to extend this approach to

a multi-class classification problem, where drones can be classified into different categories based on the type of payload they are carrying. This would require a larger and more diverse dataset with annotated images of drones carrying different types of payloads. By training the considered ResNet networks or other state-of-the-art classifiers on such a dataset, it would be possible to develop a more comprehensive and accurate payload detection system.

Secondly, the current study utilized a simulated dataset for training and testing the payload detection algorithm at the first step. While this approach provides a controlled environment for data collection and annotation, it may not fully capture the complexity and variability of real-world scenarios. The success of the DL-based approach depends on the quality and diversity of the training dataset and may be influenced by various factors such as lighting conditions and occlusions. In this study, the experimental data with one type of drone was employed and the trained network based on the simulated dataset could classify it with high accuracy even without seeing it before. Nonetheless, future work can involve collecting a real-world dataset of drone images and videos, with ground truth annotations for payload detection. This would require deploying drones equipped with cameras in various environments and recording target drones' flights while carrying different payloads. The collected dataset could then be used to evaluate the performance of the ResNet model in real-world scenarios and assess its robustness and generalization capabilities.

Moreover, the current study only focused on visual imagery as the input data for payload detection. However, other sensor inputs, such as radar or acoustic data, could be incorporated to enhance the detection capabilities of the system (even in the special case of active payloads, RF sensors can be considered as a choice for payload detection). These additional sensors could provide complementary information that can help differentiate between loaded and unloaded drones, especially in scenarios where visual cues may be limited or obscured. Future work could involve developing a multimodal payload detection system that combines information from multiple sensors to improve the overall detection performance.

Regarding drone localization, based on the concept that the location of an object cannot jump abruptly, future work can focus on using temporal information and trajectory analysis to improve the accuracy and robustness of object localization (although, the initial steps were done by using MA and Kalman filters). This approach takes into account the continuity and smoothness of object movement, which can provide accurate localization. One potential field for future research is the development of trajectory-based localization algorithms that utilize historical positional data to predict the future location of an object. By analyzing the trajectory patterns and dynamics of the object over time, predictive models can be built to estimate the next likely position. This can be achieved through

techniques such as RNN that are capable of learning and predicting sequential data.

Furthermore, the integration of multiple sensors and modalities can provide complementary information for more robust localization. Combining visual data with other sensor inputs can improve the accuracy and reliability of object localization. For instance, incorporating depth sensors, such as LiDAR and RGB-D camera, can enhance the understanding of the object's 3D motion and aid in precise localization. The fusion of multiple modalities can be achieved through sensor fusion techniques, such as extended Kalman filtering, Bayesian methods, or deep learning-based fusion models, which combine the strengths of different sensors to provide more accurate and robust localization results.

Lastly, since the observer drone is assumed to be equipped with an onboard camera, there is a possibility of obtaining a multi-view video of the target drone. In such cases, 3D reconstruction algorithms can be employed to extract an approximate 3D model of the target drone from the 2D images. This 3D model does not need to be highly accurate, but rather a rough representation is sufficient for addressing the considered problems. The approximate 3D model can be beneficial for both differentiating between loaded and unloaded drones and localization tasks. In addition to the primary 2D image, this model can be incorporated into a multi-modal regression network to estimate the distance to the target drone. By leveraging the additional information provided by the 3D model, the regression network can enhance the accuracy of distance estimation.

In both the localization and payload detection modules, a significant challenge arises when dealing with falsely detected objects identified as drones by the detector network (in this case, YOLO). As discussed in previous chapters, both modules generate outputs for each object detected by the network, which may include invalid data for falsely identified objects. Therefore, a potential area for future research lies in proposing approaches to address such cases, perhaps by providing output only when the system is confident in the detected object. In the context of future developments for the simulator, an important feature to consider extending is the capability to simulate radar at the signal level. In the current version of the simulator, radar output has been simulated in a specific scenario within the Blocks environment, and it primarily focuses on simulating the detection output corresponding to the target drone. To enhance this capability, a valuable addition would be the ability to simulate the reflected radar signal from the environment by continuously tracking the positions and types of objects within the scene in AirSim©. However, this task is not straightforward due to the need for various critical pieces of information, including the RCS of all materials present in the scene. Moreover, simulating the reflected signals from multiple scatterers in the scene, even for a single object, and efficiently managing all these reflections into a coherent received signal pose additional challenges.

Furthermore, an area for future development involves integrating clustering and tracking filters with the existing data-level simulator. This would allow for a more comprehensive and realistic simulation environment, which is crucial for testing and evaluating tracking and classification algorithms in complex and dynamic scenarios. Lastly, to enable the ability to navigate through different environments within AirSim©, a potential topic for future research involves real-time monitoring of information regarding obstacles ahead. This information could be used to dynamically adjust the flight path by implementing collision avoidance algorithms.

References

[1] Y. Zhao, X. Zhang, and F. Fioranelli, "Initial results of radar-based classification of commercial drone carrying small payloads," in *2019 International Radar Conference (RADAR)*, pp. 1–4, 2019.

[2] S. Rahman, D. A. Robertson, and M. A. Govoni, "Radar signatures of drones equipped with heavy payloads and dynamic payloads generating inertial forces," *IEEE Access*, vol. 8, pp. 220542–220556, 2020.

[3] I. Ku, S. Roh, G. Kim, C. Taylor, Y. Wang, and E. T. Matson, "UAV payload detection using deep learning and data augmentation," in *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, pp. 18–25, 2022.

[4] U. Seidaliyeva, M. Alduraibi, L. Ilipbayeva, and A. Almagambetov, "Detection of loaded and unloaded UAV using deep neural network," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, pp. 490–494, 2020.

[5] U. Seidaliyeva, M. Alduraibi, L. Ilipbayeva, and N. Smailov, "Deep residual neural network-based classification of loaded and unloaded UAV images," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, pp. 465–469, 2020.

[6] H. Stuckey, A. Al-Radaideh, L. Sun, and W. Tang, "A spatial localization and attitude estimation system for unmanned aerial vehicles using a single dynamic vision sensor," *IEEE Sensors Journal*, vol. 22, no. 15, pp. 15497–15507, 2022.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[8] P. Payeur, *Transformation*. Machine vision (course notes), University of Ottawa, 2022.

[9] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned aerial vehicles (uavs): Practical aspects, applications, open challenges, security issues, and future trends," *Intelligent Service Robotics*, vol. 16, no. 1, pp. 109–137, 2023.

[10] J.-P. Huttner and M. Friedrich, "Current challenges in mission planning systems for UAVs: A systematic review," in *2023 Integrated Communication, Navigation and Surveillance Conference (ICNS)*, pp. 1–7, IEEE, 2023.

[11] D. Fortune, H. Nitsch, G. Markarian, D. Osterman, and A. Staniforth, "Counter-unmanned aerial vehicle systems: Technical, training, and regulatory challenges," *Security Technologies and Social Implications*, pp. 122–148, 2022.

[12] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif, "Towards the unmanned aerial vehicles (UAVs): A comprehensive review," *Drones*, vol. 6, no. 6, p. 147, 2022.

[13] J. Farlík and L. Gacho, "Researching uav threat–new challenges," in *2021 International Conference on Military Technologies (ICMT)*, pp. 1–6, IEEE, 2021.

[14] M. Hassanalian and A. Abdelkefi, "Classifications, applications, and design challenges of drones: A review," *Progress in Aerospace Sciences*, vol. 91, pp. 99–131, 2017.

[15] G. Singhal, B. Bansod, and L. Mathew, "Unmanned Aerial Vehicle classification, applications and challenges: A review," *Preprints*, 2018.

[16] FAA, "Press Release – FAA Releases Unmanned Aircraft Systems Integration Roadmap." [Online]. Available: http://www.tc.faa.gov/its/worldpac/uas/UAS_Roadmap_2013.pdf, November 2021.

[17] FAA, "FAA Aerospace Forecasts." [Online]. Available: https://www.faa.gov/sites/faa.gov/files/2022-06/Unmanned_Aircraft_Systems.pdf, July 2021.

[18] IEEE, "IEEE Xplore search results for drone detection." [Online]. Available: https://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=drone%20detection.

[19] R. Jiang, Y. Zhou, and Y. Peng, "A review on intrusion drone target detection based on deep learning," in *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, vol. 4, pp. 1032–1039, IEEE, 2021.

[20] Y. Liu, P. Sun, N. Wergeles, and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection," *Expert Systems with Applications*, p. 114602, 2021.

[21] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *arXiv preprint arXiv:2104.11892*, 2021.

[22] 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*, pp. 779–788, 2016.

[23] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[24] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis," *arXiv preprint arXiv:2211.04800*, 2022.

[25] 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)*, pp. 3645–3649, IEEE, 2017.

[26] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 748–756, IEEE, 2018.

[27] B. Vergouw, H. Nagel, G. Bondt, and B. Custers, "Drone technology: Types, payloads, applications, frequency spectrum issues and future developments," *The Future of Drone Use: Opportunities and Threats from Ethical and Legal Perspectives*, pp. 21–45, 2016.

[28] C. Norman, "A global review of prison drug smuggling routes and trends in the usage of drugs in prisons," *Wiley Interdisciplinary Reviews: Forensic Science*, vol. 5, no. 2, p. e1473, 2023.

[29] H. V. Sethuraman, A. Yarovoy, and F. Fioranelli, "Classification of unmanned aerial vehicles (uavs) carrying payloads with polarimetric radar," in *2021 18th European Radar Conference (EuRAD)*, pp. 365–368, IEEE, 2022.

[30] F. Svanström, C. Englund, and F. Alonso-Fernandez, "Real-time drone detection and tracking with visible, thermal and acoustic sensors," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 7265–7272, IEEE, 2021.

[31] A. Coluccia, G. Parisi, and A. Fascista, "Detection and classification of multirotor drones in radar sensor networks: A review," *Sensors*, vol. 20, no. 15, p. 4172, 2020.

[32] F. Fioranelli, M. Ritchie, H. Griffiths, and H. Borrion, "Classification of loaded/unloaded micro-drones using multistatic radar," *Electronics Letters*, vol. 51, no. 22, pp. 1813–1815, 2015.

[33] M. Ritchie, F. Fioranelli, H. Borrion, and H. Griffiths, "Multistatic micro-doppler radar feature extraction for classification of unloaded/loaded micro-drones," *IET Radar, Sonar & Navigation*, vol. 11, no. 1, pp. 116–124, 2017.

[34] J. S. Patel, C. Al-Ameri, F. Fioranelli, and D. Anderson, "Multi-time frequency analysis and classification of a micro-drone carrying payloads using multistatic radar," *The Journal of Engineering*, vol. 2019, no. 20, pp. 7047–7051, 2019.

[35] J. J. De Wit, D. Gusland, and R. P. Trommel, "Radar measurements for the assessment of features for drone characterization," in *2020 17th European Radar Conference (EuRAD)*, pp. 38–41, 2021.

[36] S. Rahman, D. A. Robertson, and M. A. Govoni, "Radar signatures of drones equipped with liquid spray payloads," in *2020 IEEE Radar Conference (RadarConf20)*, pp. 1–5, 2020.

[37] L. Pallotta, C. Clemente, A. Raddi, and G. Giunta, "A feature-based approach for loaded/unloaded drones classification exploiting micro-doppler signatures," in *2020 IEEE Radar Conference (RadarConf20)*, pp. 1–6, 2020.

[38] H. V. Sethuraman, "Radar-based classification of Unmanned Aerial Vehicles (UAVs) carrying payloads," Master's thesis, Delft University of Technology, 2021.

[39] Y. Wang, F. E. Fagiani, K. E. Ho, and E. T. Matson, "A feature engineering focused system for acoustic UAV payload detection," in *ICAART (3)*, pp. 470–475, 2022.

[40] O. A. Ibrahim, S. Sciancalepore, and R. Di Pietro, "Noise2weight: On detecting payload weight from drones acoustic emissions," *Future Generation Computer Systems*, vol. 134, pp. 319–333, 2022.

[41] R. Kreuzig, M. Ochs, and R. Mester, "DistanceNet: Estimating traveled distance from monocular images using a recurrent convolutional neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1258–1266, 2019.

[42] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[43] J. Zhu and Y. Fang, "Learning object-specific distance from a monocular image," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3839–3848, 2019.

[44] I. White, D. K. Borah, and W. Tang, "Robust optical spatial localization using a single image sensor," *IEEE Sensors Letters*, vol. 3, no. 6, pp. 1–4, 2019.

[45] M. A. Haseeb, J. Guan, D. Ristic-Durrant, and A. Gräser, "DisNet: a novel method for distance estimation from monocular camera," *10th Planning, Perception and Navigation for Intelligent Vehicles (PPNIV18), IROS*, 2018.

[46] A. Masoumian, D. G. Marei, S. Abdulwahab, J. Cristiano, D. Puig, and H. A. Rashwan, "Absolute distance prediction based on deep learning object detection and monocular depth estimation models.," in *CCIA*, pp. 325–334, 2021.

[47] C.-C. Lo and P. Vandewalle, "Depth estimation from monocular images and sparse radar using deep ordinal regression network," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 3343–3347, IEEE, 2021.

[48] V. R. Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, and P. Mader, "Syndistnet: Self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 61–71, 2021.

[49] J. Yu and H. Choi, "YOLO MDE: Object detection with monocular depth estimation," *Electronics*, vol. 11, no. 1, p. 76, 2021.

[50] M. Vajgl, P. Hurtik, and T. Nejezchleba, "Dist-YOLO: Fast object detection with distance estimation," *Applied Sciences*, vol. 12, no. 3, p. 1354, 2022.

[51] A. Masoumian, H. A. Rashwan, J. Cristiano, M. S. Asif, and D. Puig, "Monocular depth estimation using deep learning: A review," *Sensors*, vol. 22, no. 14, p. 5353, 2022.

[52] H. Liang, Z. Ma, and Q. Zhang, "Self-supervised object distance estimation using a monocular camera," *Sensors*, vol. 22, no. 8, p. 2936, 2022.

[53] Y.-C. Lai and Z.-Y. Huang, "Detection of a moving UAV based on deep learning-based distance estimation," *Remote Sensing*, vol. 12, no. 18, p. 3035, 2020.

[54] H. Stuckey, A. Al-Radaideh, L. Escamilla, L. Sun, L. G. Carrillo, and W. Tang, "An optical spatial localization system for tracking unmanned aerial vehicles using a single dynamic vision sensor," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3093–3100, IEEE, 2021.

[55] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[56] S. Sonkar, P. Kumar, R. C. George, T. Yuvaraj, D. Philip, and A. Ghosh, "Real-time object detection and recognition using fixed-wing lale vtol uav," *IEEE Sensors Journal*, vol. 22, no. 21, pp. 20738–20747, 2022.

[57] M. Kurosaki, D. Kawaguchi, K. Ogawa, R. Nakamura, and H. Hadama, "Comparison of drone classification accuracy for cnn models using UWB radar," in *2022 Asia-Pacific Microwave Conference (APMC)*, pp. 907–909, IEEE, 2022.

[58] A. S. Mubarak, M. Vubangsi, F. Al-Turjman, Z. S. Ameen, A. S. Mahfudh, and S. Alturjman, "Computer vision based drone detection using mask R-CNN," in *2022 International Conference on Artificial Intelligence in Everything (AIE)*, pp. 540–543, IEEE, 2022.

[59] W. Meng and M. Tia, "Unmanned aerial vehicle classification and detection based on deep transfer learning," in *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, pp. 280–285, IEEE, 2020.

[60] H. M. Oh, H. Lee, and M. Y. Kim, "Comparing convolutional neural network (CNN) models for machine learning-based drone and bird classification of anti-drone system," in *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, pp. 87–90, IEEE, 2019.

[61] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[62] Autodesk Inc., "Autodesk Maya." [Online]. Available: https://www.autodesk.ca/en/products/maya/overview/.

[63] SZ DJI Technology Co., Ltd., "DJI Official Website." [Online]. Available: https://www.dji.com/.

[64] Epic Games Inc., "Unreal Marketplace." [Online]. Available: https://www.unrealengine.com/marketplace/en-US/.

[65] Z. Xu, X. Zhan, B. Chen, Y. Xiu, C. Yang, and K. Shimada, "A real-time dynamic obstacle tracking and mapping system for uav navigation and collision avoidance with an rgb-d camera," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10645–10651, IEEE, 2023.

[66] Z. Lin, W. Xu, and W. Wang, "A moving target tracking system of quadrotors with visual-inertial localization," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3296–3302, IEEE, 2023.

[67] M. de Rochechouart, B. I. Ahmad, A. E. F. Seghrouchni, F. Barbaresco, S. Harman, and R. A. Zitar, "Drone tracking based on the fusion of staring radar and camera data: An experimental study," in *2023 IEEE Radar Conference (RadarConf23)*, pp. 01–06, IEEE, 2023.

[68] L. Huang, T. Zhe, J. Wu, Q. Wu, C. Pei, and D. Chen, "Robust inter-vehicle distance estimation method based on monocular vision," *IEEE Access*, vol. 7, pp. 46059–46070, 2019.

[69] X. Zhang, L. Zhang, D. Xu, and H. Pei, "Multi-loss function for distance-to-collision estimation," in *2021 8th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, pp. 205–210, IEEE, 2021.

[70] S. J. Julier and J. K. Uhlmann, "Consistent debiased method for converting between polar and cartesian coordinate systems," in *Acquisition, Tracking, and Pointing XI*, vol. 3086, pp. 110–121, SPIE, 1997.

[71] M. L. Hornick, *GPS Coordinate conversion*. Technical report, Milwaukee School of Engineering, 2020.

[72] T. A. Kesury, *RADAR Modeling For Autonomous Vehicle Simulation Environment using Open Source*. PhD thesis, Purdue University, Indianapolis, Indiana, US, May 2022.

[73] J. Giovagnola, J. B. M. Megías, M. M. Fernández, M. P. Cuéllar, and D. P. M. Santos, "AirLoop: A simulation framework for testing of UAV services," *IEEE Access*, vol. 11, pp. 23309–23325, 2023.

APPENDICES

Appendix A

Instruction of using sensor-fusion simulator

A.1 Introduction

As discussed in Section 4.5.2, a GUI is designed for the sensor fusion simulator, allowing users to define their desired scenarios and customize their parameters. This appendix provides detailed instructions on how to use the GUI and outlines the necessary steps to execute the scenario within the AirSim© environment.

A.2 Various sections of the GUI

Various sections of the GUI are identified in Figure A.1. This figure illustrates that users have the capability to draw custom flight paths for each drone and adjust their parameters using the corresponding interface. Additionally, on the left side of the GUI, users can fine-tune radar parameters. Finally, there are buttons available to generate the necessary codes for running the scenario and recording data within AirSim©. The following provides a detailed explanation of each section:

- **Draw Flight Path and Set Parameters**: To commence the simulation, users must sketch the desired flight path for each drone. This is accomplished by continuously pressing the left mouse button to create the path. In the AirSim© coordinate frame, the X and Y axes correspond to the vertical and horizontal axes of the canvas,

respectively. After drawing each path, a pop-up window will appear, enabling users to specify the altitude and drone type. Since the canvas represents a 2D environment, users can independently set the drone's altitude in this pop-up window. To do so, both the initial and final altitudes for the drone are defined. The final altitude value for each point along the flight path is determined through linear curve fitting between these two altitude values, as expressed by (A.1):

$$Alt_k = Alt_{initial} + k * \frac{Alt_{final} - Alt_{initial}}{N_{points} - 1} \tag{A.1}$$

In this equation, $Alt_{initial}$ and Alt_{final} represent the initial and final altitudes of the drone on the flight path, respectively. Alt_k signifies the altitude of the kth point on the path, where k ranges from 0 to $N_{points} - 1$. Finally, N_{points} denotes the total number of points on the flight path.

- **Setting Radar Parameters**: Users can configure radar parameters, including specifying its XYZ coordinates for positioning and determining the azimuth and elevation antenna beamwidth, which defines the radar's coverage area. The number of detection instances for each drone along its flight path is determined by the user-defined detection probability. Once the radar settings are applied, the radar's position will be visually represented by a red dot on the canvas, and its coverage area will be displayed as a blue region.

- **Preparing Required Python Codes**: After configuring various scenario parameters, such as the flight path for each drone and their types, users need to prepare several Python codes (along with a *json* file) to run the scenario in AirSim©. These files include:

  - **Settings File**: This file allows users to adjust various settings, such as drone specifications, camera configurations, and general AirSim© simulation parameters like clock speed. The "Settings" button on the GUI generates the required *settings* file in the AirSim© subfolder on the computer (e.g., *./Documents/AirSim/* on Windows).
  - **Flight Controller Code**: This Python code, executed after starting AirSim©, controls the drones to follow their designated flight paths. Once again, the corresponding button on the GUI generates this file in the AirSim© subfolder on the computer.
  - **Detection Code**: To utilize AirSim© "Object Detection" feature and extract bounding box (BB) data as well as XYZ coordinates of each drone in

the recorded scene, a Python code should be run simultaneously while running AirSim©. This code is saved in the AirSim© subfolder.

– **Radar Simulator Code**: The corresponding button generates a Python code called "RadarSimulatorCode" in the same local folder where the GUI code is saved. This Python code simulates the radar detection output for each drone within the radar's coverage area.

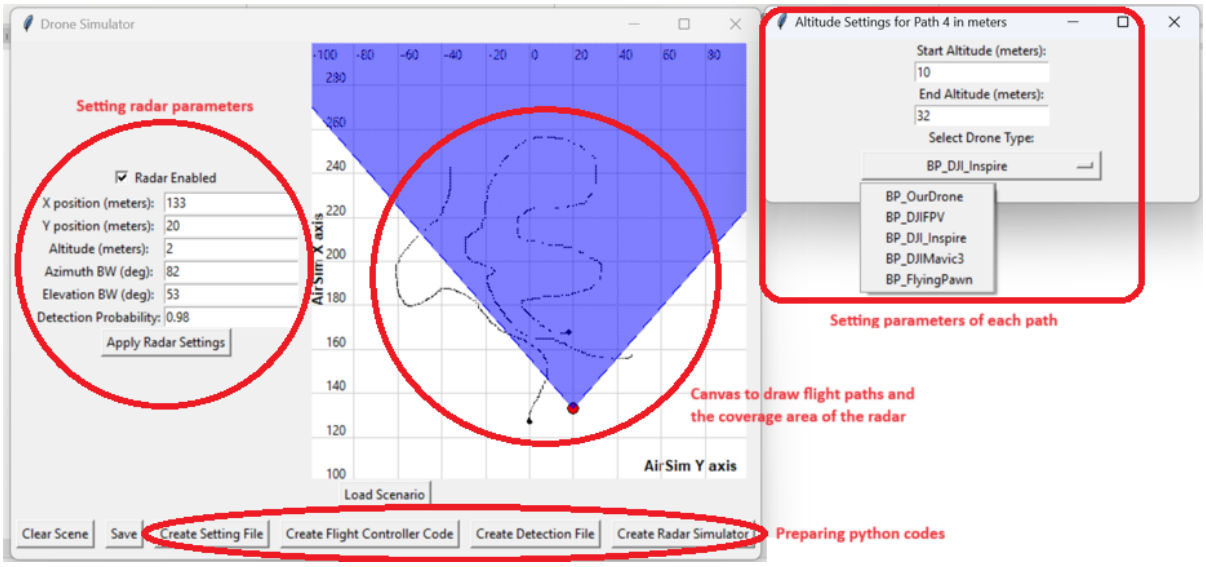


Figure A.1: Different parts of the GUI of the developed simulator

In addition to the aforementioned buttons, there are three other buttons available on the GUI:

- **Clear scene**: Pressing this button removes all drawn flight paths and the plotted radar coverage area from the scene, allowing the user to start a new scenario.
- **Save**: This option saves the XYZ coordinates of the drones' flight paths. The data is stored as a *text* file named "flight_plan.txt" in the same directory as the GUI. The format of the file is as follows:

```
Curve ID: 1
X: 200.99009900990097, Y: 57.92079207920793, Z: -1.0
X: 200.0, Y: 58.415841584158414, Z: -1.0416666666666667
```

```
...
Curve ID: 2
X: 198.5148514851485, Y: -54.45544554455446, Z: -22.0
X: 197.5247524752475, Y: -52.97029702970297, Z: -22.0
...
```

where the XYZ coordinates of the points for each curve are listed after the corresponding "Curve ID." It is important to note that all flight paths are interpolated to have the same number of points for implementation purposes in AirSim©.

- **Load scence**: This option allows the user to import a *text* file containing the XYZ coordinates of the flight paths for the desired number of drones. The format of the *text* file should match the structure presented in the previous point.

A.3 Using the simulator

The steps for using the simulator are outlined in Algorithm 1.1. These steps involve initializing the scenario within the GUI and subsequently running AirSim© to execute the desired scenario.

1.1 Steps of using the simulator

1: Draw the desired flight path for each drone and set its parameters **Or** import the scenario *text* file
2: Enable the radar and adjust its parameters (and *apply* the changes)
3: Save the scenario
▷ Save the flight paths in a text file for future reference and for use in subsequent steps
4: Create the flight controller code
▷ The code that controls the drones in the AirSim environment
5: Create the detector code
▷ The code that saves the bounding box and location information of each drone in AirSim
6: Create the radar simulator code
▷ Generating the code that simulates the radar output
7: Running AirSim
8: A new command prompt window and running the flight controller code
9: A new command prompt window and running the detector code
10: Start the scenario in AirSim by pressing the "Play" button
11: Running the radar simulator code on the saved outputs from AirSim