Counter-Drone Technology: A Drone-Based System for Detecting and Neutralizing Unmanned Aerial Vehicles

A final capstone report submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMPUTER ENGINEERING

by

SWAYANSU BARAL	(102015028)
DIVYANSH KALIA	(102015037)
ARJAN SINGH	(102015047)
SMRIDDHI SAHNI	(102015044)
RITIKA GODARA	(102065011)

under the supervision of

Dr. Amit Mishra

Dr. Karun Verma

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY, PATIALA

December 2023

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/-data/fact/source in our submission to the best of our knowledge. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature

Swayansu Baral
102015028

Place: Patiala
Date: 18-12-2023

Signature

Divyansh Kalia
102015037

Signature

Arjan Singh
102015047

Signature

Smriddhi Sahni

102015044

Signature

Ritika Godara

102065011

CERTIFICATE

This is to certify that the report titled COUNTER DRONE TECHNOLOGY PROJECT REPORT, submitted by Swayansu Baral, Divyansh Kalia, Arjan Singh Sandhu, Smriddhi Sahni, and Ritika Godara to Thapar Institute of Engineering & Technology, Patiala, for the award of the degree of Bachelor of Technology, is a record of the project work done by them under our supervision. The contents of this report, in whole or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Patiala
Date: 18-12-2023

Signature
Dr. Amit Mishra
Designation

Signature
Dr. Karun Verma
Designation

Supervisors
Department of Electronics and Communication Engineering

TIET Patiala - 147004

ACKNOWLEDGMENTS

Thanks to all those who helped us during your thesis and research work.

We would like to thank our mentor(s), Dr. Amit Mishra and Dr. Karun Verma. They have been of great help in our venture and an indispensable resource of technical knowledge. They are genuinely amazing mentors to have.

We are also thankful to our friends who devoted their valuable time and helped us in all possible ways towards the successful completion of this project. We thank all those who have contributed either directly or indirectly to this project.

Lastly, we would also like to thank our families for their unyielding love and encouragement. They always wanted the best for us, and we admire their determination and sacrifice.

ABSTRACT

As the use of drones for military applications increases, so does the threat of hostile drones being deployed against military installations, personnel, and equipment. The need for an anti-drone system that can detect, track, and neutralize unauthorized drones has become critical to ensure the safety and security of military operations.

However, the development of an effective anti-drone system that can be deployed in a military context is complex and challenging. Many existing anti-drone systems rely on fixed installations or ground-based equipment, limiting their mobility and effectiveness. Additionally, these systems often require significant resources to operate and maintain, making them expensive and difficult to scale for use in the field.

To address these challenges, an innovative solution is to develop an anti-drone system that uses a drone itself as the platform for detection and neutralization. This approach could provide the necessary mobility and flexibility to operate in a variety of settings, while also reducing the need for costly ground-based equipment. Therefore, the problem statement for this project is to design, develop, and implement an anti-drone system that uses a drone as the primary platform for military applications, which is capable of detecting and neutralizing hostile drones while minimizing collateral damage and ensuring the safety of military personnel and infrastructure.

TABLE OF CONTENTS

			ACKNOWLEDGEMENTS	i
			ABSTRACT	ii
			LIST OF FIGURES	v
			LIST OF TABLES	vi
			ABBREVIATIONS	vii
			NOTATION	viii
1.			INTRODUCTION	
	1.1		Project Overview	1
	1.2		Motivation	3
	1.3		Assumptions and Constraints	4
	1.4		Novelty of Work	6
2.			LITERATURE SURVEY	
	2.1		Research Papers	8
		2.1.1	Research Paper 1	8
		2.1.2	Research Paper 2	8
		2.1.3	Research Paper 3	8
		2.1.4	Research Paper 4	9
		2.1.5	Research Paper 5	9
		2.1.6	Research Paper 6	9
		2.1.7	Research Paper 7	9
		2.1.8	Research Paper 8	10
	2.2		Research Gaps	10
	2.3		Problem Definition	11
	2.4		Scope	11

3.		PROBLEM FORMULATION AND OBJECTIVES	12
4.		PROJECT DESIGN AND DESCRIPTION	
	4.1	Methodology	13
	4.2	Results	16
5.		OUTCOMES AND PROSPECTIVE LEARNING	34
6.		PROJECT TIMELINE	
	6.1	Project Timeline	37
	6.2	Individual Plan	39
7.		CONCLUSION AND FUTURE WORK	44
		REFERENCES	
		APPENDIX	47

LIST OF FIGURES

Sno.	Figure Name	Section
1.	Project Flow Chart	1.1
2.	Drone Model	4.1
3.	Integration Testing	4.1
4.	Sensor Integration	4.1
5.	Yolo v7 code Snippet-1	4.2
6.	Yolo v7 code Snippet-2	4.2
7.	Testing output-1	4.2
8.	Testing output-2	4.2
9.	Testing output-3	4.2
10.	GPS Module	4.2
12.	RP LIDAR	4.2
13.	Pixhawk with RC receiver	4.2
14.	Jetson nano	4.2
15.	Intel Realsense Depth Camera	4.2
16.	Drone with complete sensors	4.2
17.	Lidar in Working	4.2

LIST OF TABLES

Sno.	Table Name	Section
1.	Gantt Chart	6.1
2.	Individual Plan:	6.2
3.	Swayansu Baral	6.2
4.	Divyansh Kalia	6.2
5.	Arjan Singh Sandhu	6.2
6.	Smriddhi Sahni	6.2
7.	Ritika Godara	6.2

ABBREVIATIONS

TIET Patiala	Thapar Institute of Engineering & Technology,
API	Application Programming Interface
CV	Computer Vision
DDCS	Drone Detection and Classification System
FOV	Field of View
FPS	Frames Per Second
GDPR	General Data Protection Regulation
GPS	Global Positioning System
IoT	Internet of Things
IP	Intellectual Property
LiDAR	Light Detection and Ranging
ML	Machine Learning
MMWARS	Millimeter-Wave Automotive Radar Sensors
OCV	OpenCV (Open Source Computer Vision Library)
R&D	Research and Development
RCS	Radar Cross-Section
RF	Radio Frequency
ROI	Region of Interest
SDK	Software Development Kit
UAS	Unmanned Aircraft Systems
YOLO v7	You Only Look Once version 7

NOTATION

The research scholar/student must explain the meaning of special symbols and notations used in the thesis. Define English symbols, Greek symbols, and then miscellaneous symbols Some examples are listed here.

r	density, $\frac{kg}{m^3}$
r	Radius, m
q	Angle between x and y in degrees
v	velocity of the object

CHAPTER 1

INTRODUCTION

1.1 Project Overview

The main objective of this project is to develop a comprehensive system for the detection, location, and classification of Unmanned Aerial Systems (UAS) or Drones. The system integrates advanced technologies such as a camera, Lidar sensor, OpenCV, and YOLO v7 (You Only Look Once, version 7) to detect drones in real time and accurately in different environments.

Traditionally, detection technologies have faced limitations, especially in adverse weather conditions or environments with significant clutter. To address these challenges, our project focuses on developing an innovative and comprehensive drone detection system that leverages a combination of a camera, LIDAR sensor, OpenCV, and YOLO v7 (You Only Look Once version 7). This integrated approach aims to overcome the shortcomings of traditional methods and provide a robust solution for real-time drone identification in diverse scenarios.

- **Camera-based detection with YOLO v7:**

  The YOLO v7 model is used for efficient and real-time object detection using visual data from the camera. This component focuses on detecting and locating drones in various environments.

- **Lidar sensor for measuring distance and object angle:**

  The Lidar sensor measures the distance between the sensor and the detected drones. This information helps pinpoint drones in the surrounding space.

- **Machine learning algorithms for integration:**

  Machine learning algorithms adapted to OpenCV image analysis help integrate data from different sensors. These algorithms are trained on a rich dataset containing information about targets detected by YOLO, LIDAR sensor measurements and RF inspection.

- **Performance of YOLO v7:**

  YOLO v7 is expected to enable accurate and real-time detection of drones in various scenarios. Computing Requirements: It is assumed that the available computing power can efficiently handle the processing requirements of YOLO v8 and OpenCV.

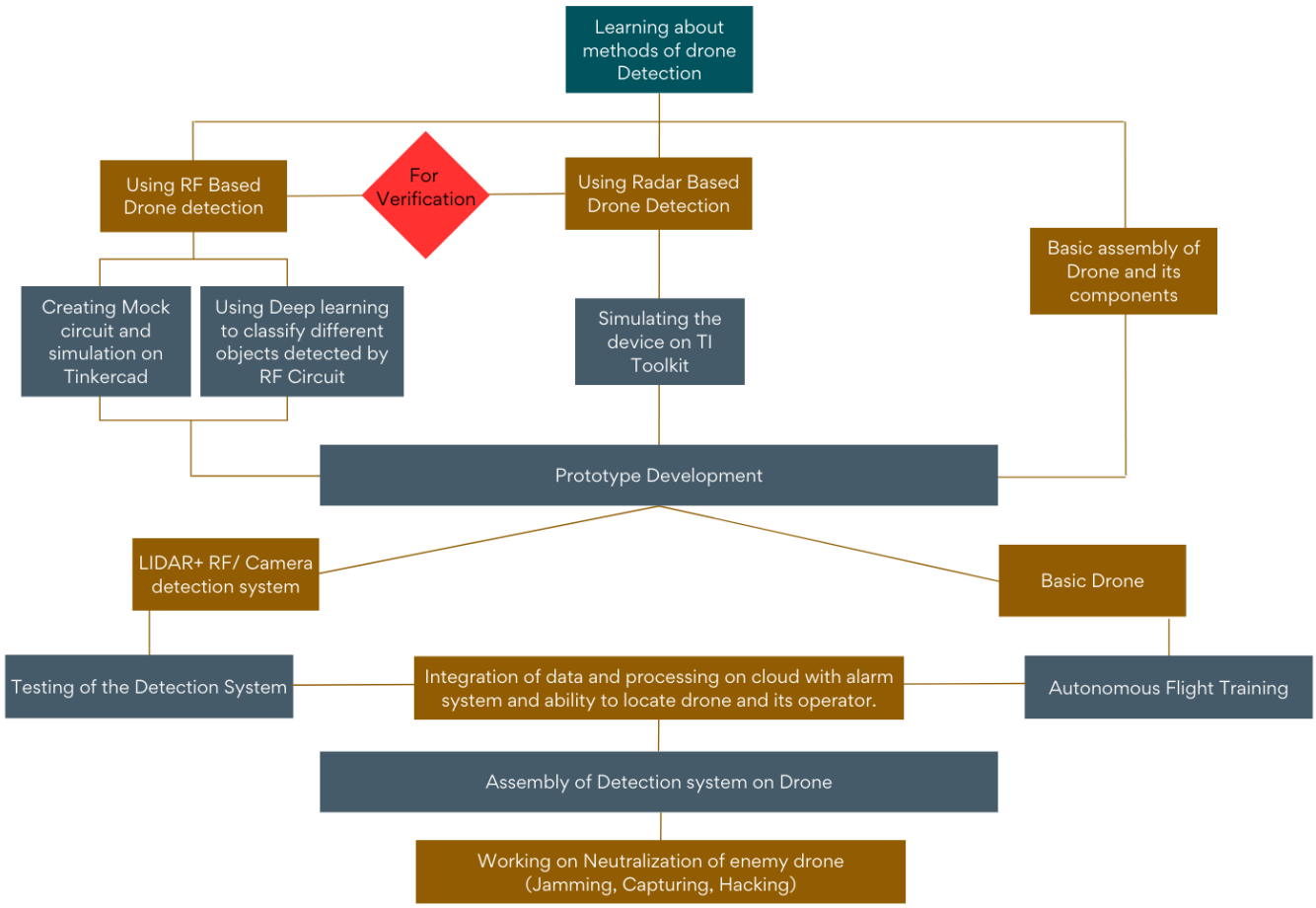

Fig. 1- Project Flow Chart

1.2 Motivation

Emerging challenges in drone technology:

The proliferation of Unmanned Aerial Systems (UAS) or drones in various sectors has led to changes in areas such as entertainment, business, and infrastructure. However, the rapid growth of drone technology has also brought safety, security, and privacy challenges.

Drones with versatile applications have become available to hobbyists, commercial users, and potentially malicious actors. This accessibility has raised concerns about unauthorized drone activity threatening critical infrastructure, public safety, and privacy.

Limitations of traditional identification methods:

Traditional drone detection methods, including optical vision and audio sensors, often face limitations, especially in adverse weather conditions or environments with high clutter. These methods can struggle to provide reliable and real-time drone detection, especially in dynamic and complex scenarios.

The need for innovative identification and classification systems:

The motivation for this project stems from the need to address the limitations of existing drone detection technologies. The goal is to create an advanced and comprehensive system capable of effectively detecting, locating, and classifying drones in real-time in various environments.

Use of advanced technologies:

The integration of a camera, LIDAR sensor, OpenCV, and YOLO v7 represents a futuristic approach. Utilizing computer vision, deep learning, and sensor technologies, the project aims to overcome the shortcomings of traditional methods and provide a reliable solution that works under changing conditions.

Contribution to security:

Motivation goes beyond technological innovation; this aligns with the broader goal of increasing safety, security and overall well-being. The purpose of the proposed system is to contribute to the protection of critical infrastructure, the prevention of unauthorized drone activity and the provision of timely information for threat assessment and response.

Bridging the Gaps in Drone Detection Research:

This project is motivated by a commitment to fill existing research gaps in the field of drone detection. By exploring new combinations of sensors and technologies, the goal is to contribute to developing more robust and versatile drone detection and classification systems.

Overall effect:

The ultimate motivation is to create a concrete and impressive solution that meets the challenges of the evolving landscape of drone technology. The project and its results are expected to have applications in several fields, including security and surveillance, contributing to the responsible and safe integration of drones into our society.

1.3 Assumptions and Constraints

This section describes the assumptions and constraints of the proposed solution.

Assumptions:

Sensor Availability:

The mission assumes the equipped availability and capability of the digital digicam and LIDAR sensor components. It is presumed that those sensors can efficiently seize visible information and degree distances, respectively.

YOLO v8 Performance:

Assumption that YOLO v7 will offer correct and real-time drone detection. The effectiveness of the YOLO version is a key assumption for the achievement of the mission.

Computational Resources:

The mission assumes the provision of enough computational sources able to managing the processing needs of YOLO v7, OpenCV, and gadget mastering algorithms for real-time operation.

Stable RF Signatures:

The assumption is made that RF signatures related to one of a kind drone sorts continue to be strong and regular over time, bearing in mind dependable RF verification.

Operational Environment:

The mission assumes operational surroundings with managed situations, along with line-of-sight operation for the sensors. It is presumed that drones will function inside the powerful area of view of the sensors.

Prior Knowledge of Drone Signatures:

Assumption that there's previous know-how or a database of drone signatures, along with RF traits. This record is important for distinguishing drones from different items detected with the aid of using the sensors.

No Signal Interference:

The mission assumes that the overall performance of the drone detection and class machine may be unaffected by using minimum to no interference from different radar structures or outside sources. It presumes that the electromagnetic surroundings is essentially unpolluted.

Controlled Environmental Conditions:

The mission assumes that the operational surroundings is relatively strong and beneath under control, freed from intense climate activities or different factors that may adversely have an affect on the functioning of the sensors. It is thought that those situations may be maintained inside the environments in which the machine could be used maximum frequently.

Computational Power:

Assumption that sufficient computational strength is required for the mission to method the sensor information, run sign processing algorithms, and perform the drone class algorithms in real-time. It is thought that the processing strength to hand can take care of the machine's complexity and information volume.

Drone Signatures:

The mission assumes that there's previous know-how or a database of drone signatures, along with their radar cross-section (RCS) traits or different distinguishing features. This record is used to distinguish drones from different items or litter detected using the sensors.

These assumptions together shape the premise for the design, implementation, and operation of the drone detection and class machine. It is essential to not forget those assumptions at some stage in the mission lifecycle for sensible making plans and powerful machine overall performance.

Constraints:

Cost limits:

The project is subject to financial restrictions, including restrictions on the purchase of sensors, hardware, and software components. The system must be cost-effective and meet budget constraints.

Time limits:

Strict time limits, including deadlines for project phases (design, implementation, testing), can affect the project and its duration. Effective time management is essential to achieving project milestones.

Regulatory compliance:

The project must comply with all relevant rules and regulations regarding using sensors, radio frequency technology, and drone detection systems. Compliance with local, national and international laws is a constraint that must be addressed.

Technical limitations:

Technical limitations may arise from the strengths and weaknesses of the selected sensors, the YOLO v7 model and OpenCV. Factors such as sensor sensitivity, range, and processing capabilities can limit overall system performance.

Environmental factors:

The project may face environmental constraints, including weather conditions and disturbances that may affect the sensors and #039; presentation Adverse weather or environmental disturbances must be considered and minimized as much as possible.

Scalability Limits:

The project may have limitations on scalability, especially if the system is intended for use in larger areas or locations. The performance and computational requirements of the system should be scalable to accommodate more sensors, larger data volumes, and processing requirements associated with detecting and classifying multiple drones simultaneously. These limitations present challenges that must be carefully navigated during design and development. Mitigating these effects and finding optimal solutions within these constraints are essential for the successful implementation and deployment of a drone detection and classification system.

1.4 Novelty of Work

Integration of YOLO v7 for Real-Time Object Detection:

The challenge introduces novelty through the mixing of YOLO v7 (You Only Look Once model 7), a modern-day item detection model. YOLO v7 is famous for its performance and speed, allowing real-time detection of drones. This integration represents a modern method to rapidly and correctly discover drones, distinguishing the challenge from conventional detection methods.

Adapting OpenCV for RF Verification:

The challenge innovates through adapting OpenCV, a strong PC imaginative and prescient library, for Radio Frequency (RF) verification. OpenCV is hired to investigate RF signatures related to extraordinary drone sorts captured through the digital digicam. This integration complements class accuracy by combining visible and RF information, showcasing the adaptability and flexibility of the gadget.

Comprehensive Data Integration with Machine Learning Algorithms:

A novel factor of the challenge lies inside the integration of information from more than one sensor—visible information from YOLO v7, distance measurements from the LIDAR sensor, and RF verification from OpenCV—the usage of gadget getting to know algorithms. These algorithms, tailored for picture analysis, facilitate the seamless mixture of various information sources, contributing to a holistic and complete method to drone detection and class.

Real-Time Operation in Diverse Environments:

The integration of YOLO v8 and the adaptive use of OpenCV for RF verification make contributions to the gadget's capacity to function in real-time throughout various environments. This functionality complements the responsiveness and adaptableness of the gadget, making it perfect for dynamic situations and extraordinary operational situations.

Reduction of False Positives and False Negatives:

This challenge advances the sphere of drone detection through in particular addressing the undertaking of decreasing fake positives and fake negatives. The improvement makes a specialty of refining algorithms and methodologies to beautify the accuracy of drone identification. By doing so, the gadget goals to offer extra dependable facts for danger evaluation and response.

Transition from Radar to Camera and LIDAR Sensor:

A sizable shift in sensor technology—from millimeter-wave radar sensors to a mixture of a digital digicam and LIDAR sensor—provides a layer of novelty to the challenge. This transition displays an adaptive method to sensor selection, catering to precise challenge necessities and probably increasing the gadget's applicability to various situations.

Versatile System for Various Environmental Conditions:

The challenge introduces versatility by incorporating sensors that may function efficaciously in extraordinary environmental situations. The mixture of a digital digicam and LIDAR sensor allows the gadget to feature in numerous settings, overcoming barriers posed by climate situations and environmental interference.

Addressing Research Gaps in Drone Detection:

The studies contribute to filling present gaps within the subject of drone detection, especially within the context of millimeter-wave radar-primarily based total identification. The challenge's exploration of novel sensor combos and technology serves to improve the modern know-how of powerful and dependable drone detection systems.

Applications in Security, Surveillance, and Safety:

Beyond technical innovation, the challenge's novelty extends to its ability packages in vital regions along with security, surveillance, and protection. The effects of the challenge goal to make contributions to the safety of key infrastructure, public spaces, and the general protection of various environments in which drone sports may also pose challenges.

The mixture of those novel factors positions the challenge as a forward-searching and complete answer within the area of drone detection, class, and localization.

CHAPTER 2

LITERATURE SURVEY

2.1 RESEARCH PAPERS

2.1.1 Research Paper 1

- Literature Analysis: **Drone Detection and Classification Based on Radar Cross Section Signatures**
- Leveraging RCS signatures for drone detection and classification: The work utilizes a database of RCS signatures to detect and classify drones. Measurement results of a carbon fiber drone's RCS at 28 GHz provide essential information about its signature.
- Assessing detection probability and range error: Simulations in urban environments evaluate the effectiveness of RCS-based detection. Drones at distances of 30m to 90m use experimentally obtained RCS signatures for accurate detection and classification.
- Significance of experimental RCS signatures: The study highlights the importance of experimentally derived RCS signatures in enhancing the accuracy and reliability of real-world drone detection systems.

2.1.2 Research Paper 2

- Literature Analysis: **Deep Learning for RF-Based Drone Detection and Identification : A multi-channel 1-D Convolutional Neural Networks Approach**
- Growing popularity of commercial drones creates security concerns for sensitive areas like airfields and military bases, emphasizing the need for drone detection and identification systems.
- The paper proposes a deep learning-based approach using a multi-channel 1-dimensional convolutional neural network to detect and identify drones, leveraging a publicly available database of drone's radio frequency signals.
- The proposed model produces compact feature representations of the dataset, enabling the use of classical machine learning algorithms for efficient classification, enhancing the effectiveness of the drone detection and identification system

2.1.3 Research Paper 3

- Literature Analysis: **Deep Learning Inspired Vision Based Frameworks For Drone Detection**
- Drone detection technology is critical for defense systems due to increasing crimes and terrorist attacks involving commercial drones, requiring prompt responses from law enforcement agencies.
- The lack of benchmark datasets and performance metrics for drone detection poses a challenge, hindering the comparison and evaluation of different detection models.
- The paper presents three effective single shot detectors based on YOLOv4, YOLOv5, and DETR architectures, achieving impressive results with maximum average precision (AP) of 99% and average Intersection over Union (IOU) of 84%. The precision-recall curves validate the generalization and fitness of these models in detecting and tracking drones.

2.1.4 Research Paper 4

- Literature Analysis: **Empirical Path-Loss Modelling and RF Detection Scheme For Various Drones**
- Wi-Fi and Bluetooth signals are identified with the singular-value decomposition (SVD) algorithm by using the hopping characteristics
- General and drone Wi-Fi signals are separated by in-phase/quadrature (I/Q) phase analysis over the measurement time. The windowed received signal strength indicator (RSSI) moving detection (WRMD) analysis identifies the drone Bluetooth signal according to the movement of the drone
- The model is verified by a ray-tracing simulation. Model provides a simple and accurate prediction for designing future aerial communications systems according to changes in drone movement.

2.1.5 Research Paper 5

- Literature Analysis: **Drone Presence Detection By The Drone's RF Communication**
- Detect remote control signal: Hardware detection, based on back-end signal processing(great effect and low false alarm probability). which mainly includes the detection method based on power spectrum cancellation, autocorrelation detection and remote control signal extraction method based on spectrum.
- RF (passive): Drone communication protocols usually use the same frequency bands used for WiFi transmissions. A drone equipped with a camera usually transmits a video stream to its control unit through the same wireless channel.
- A simple method is to monitor a wide range of RF signals, such as 1 MHz to 6.8 GHz, and treat any transmitter of unknown RF signals as a drone. RF detection is detected by remote control signals or map transmission signals, but when the drone uses GPS navigation to fly autonomously, this method will fail.

2.1.6 Research Paper 6

- Literature Analysis: **Radio Frequency Toolbox For Drone Detection And Classification**
- In other to detect the signal, implement energy detection of a wireless signal.
- This processing involves the labeling and training of the collected features. This is necessary because we will be using the supervised learning method of machine learning to classify the drone signals. GNU radio blocks will be used to implement the detection, extraction, classification and testing
- GNU Radio is an open source toolkit which has been one of the frameworks for SDRs

2.1.7 Research Paper 7

- Literature Analysis: **Drone Detection and Tracking Based on Phase-Interferometric Doppler Radar**
- Increasing threats and research efforts in drone detection: The surge in copter drone usage has prompted concerns about unauthorized activities and risks, leading to research in various detection methods including acoustics, cameras, cascades, radio frequency, shooting, and netting.
- Radar systems for copter drone detection: This paper focuses on radar systems as an effective approach for detecting copter drones. It reviews existing efforts, emphasizing micro-Doppler analysis as a preferred radar technique. It introduces passive forward scatter radar (PFSR) as an emerging methodology with reported achievements in detecting ground and airborne targets.

• Micro-Doppler analysis and DVBS-based PFSR: The paper discusses parameters relevant to Doppler and micro-Doppler analysis to enhance the detection of low-profile copter drones. It concludes by proposing DVBS-based PFSR as a feasible method for drone detection, leveraging improved capabilities and micro-Doppler analysis.

2.1.8 Research Paper 8

• Literature Analysis: **Detection and Classification of Multi-Rotor Drones In Radar Sensor Network**
• Emergence of new threats and challenges: The availability of low-cost, small drones has led to unprecedented applications but also raised concerns about potential misuse, such as drug smuggling and terrorist attacks. This paper addresses the challenges of drone identification, including detection, verification, and classification.
• Importance of spatially-distributed sensor networks: Modern surveillance systems rely on spatially-distributed sensor networks for comprehensive coverage. The paper highlights the role of cost-effective and robust frequency modulated continuous wave (FMCW) radar sensors, which can operate over long distances and withstand varying environmental conditions.
• Literature review on identification approaches: The paper provides a comprehensive review of research on drone identification methods. It covers the detection of drones, target verification, and classification, contributing to the advancement of knowledge in this field.

2.2 RESEARCH GAPS

● Radar System Limitations: The research papers use millimetres-wave radar sensors for detecting and localizing UAVs, but these sensors have limitations in terms of range, resolution, and accuracy. This can affect the performance of the system in detecting and localizing small UAVs that are flying at low altitudes.
● False Alarms: The system may generate false alarms due to interference from other objects such as birds, insects, and even trees, which can be mistakenly detected as UAVs. This can lead to unnecessary alerting and cause confusion for the operators.
● Complex Signal Processing: The signal processing required to detect and localize UAVs using radar sensors is complex and requires significant computational power. The performance of the system needs to be validated in real-world scenarios with varying environmental conditions, such as wind, rain, and fog.

2.3 PROBLEM DEFINITION

- As the use of drones for military applications increases, so does the threat of hostile drones being deployed against military installations, personnel, and equipment. The need for an anti-drone system that can detect, track, and neutralize unauthorized drones has become critical to ensure the safety and security of military operations.
- However, the development of an effective anti-drone system that can be deployed in a military context is complex and challenging. Many existing anti-drone systems rely on fixed installations or ground-based equipment, limiting their mobility and effectiveness. Additionally, these systems often require significant resources to operate and maintain, making them expensive and difficult to scale for use in the field.
- To address these challenges, an innovative solution is to develop an anti-drone system that uses a drone itself as the platform for detection and neutralization. This approach could provide the necessary mobility and flexibility to operate in a variety of settings, while also reducing the need for costly ground-based equipment. Therefore, the problem statement for this project is to design, develop, and implement an anti-drone system that uses a drone as the primary platform for military applications, which is capable of detecting and neutralizing hostile drones while minimizing collateral damage and ensuring the safety of military personnel and infrastructure.

2.4 SCOPE

- Design and develop a drone-based anti-drone system that can detect, track, and neutralize unauthorized drones within a designated area of operation. Implement advanced sensors and detection technologies, such as radar, thermal imaging, and computer vision, to enhance the system's capabilities and accuracy.
- Incorporate machine learning algorithms to improve the system's ability to identify and differentiate between friendly and hostile drones in real-time. Develop an effective and safe neutralization mechanism that can disable or take control of hostile drones without causing harm to nearby people or infrastructure.
- Ensure the anti-drone system can be rapidly deployed and operated by military personnel, including intuitive control interfaces and easy-to-understand feedback.
- Conduct rigorous testing and evaluation to validate the system's performance and reliability under different weather conditions, terrain types, and drone configurations.
- Incorporate cybersecurity measures to prevent unauthorized access to the system and ensure its resilience against cyber attacks.

CHAPTER 3

PROBLEM FORMULATION AND OBJECTIVES

Problem Statement:

In the modern era, the proliferation of unmanned aircraft systems (UAS), commonly known as drones, has brought numerous benefits across recreational, commercial, and other potential domains. However, this rise in drone usage also poses significant security and safety challenges, particularly the need for robust systems capable of effective detection, localisation, and classification of drones to mitigate potential threats. Traditional drone detection methods, including optical and auditory sensors, are significantly hindered by adverse weather conditions, thereby limiting their range, precision, and overall performance.

The advent of millimeter-wave automobile radar sensors presents a promising solution, offering high resolution, extensive field of view, and most importantly, the ability to operate reliably in harsh weather conditions. Despite their advantages, there remains a research gap in leveraging these sensors for the specific application of drone detection and classification.

The primary problem this project seeks to address is the development of a reliable, precise system for drone detection, localisation, and classification using millimeter-wave radar technology. This system aims to overcome the limitations of existing detection methods by offering a solution that remains operational across diverse environmental conditions. Furthermore, the integration of RF verification adds a layer of accuracy and security, especially in differentiating between drone types such as small hobbyist drones, commercial drones, and those that may pose hostile threats.

The secondary challenge involves testing and enhancing the sensor's capabilities to discern between these various drone types, thereby improving threat assessment and response strategies. The ultimate objective is to facilitate the creation of advanced drone detection and classification systems. These systems are crucial for the protection of critical infrastructure, public safety, event security, and other relevant applications in an increasingly drone-populated environment. Thus, the project aspires to bridge the existing research gap in the field of millimeter-wave radar-based drone detection and classification, ensuring enhanced security and safety in various settings.

CHAPTER 4

PROJECT DESIGN AND DESCRIPTION

4.1 Methodology

Phase 1: Drone Hardware Development

- Design Specification: Begin by establishing the specifications for the drone hardware. This includes size, weight, payload capacity, flight time, control range, and any other relevant parameters.
- Component Selection: Choose the components such as motors, propellers, battery, flight controller, and sensors, ensuring they meet the design requirements.
- Simulation: Run simulations to test the aerodynamics, control stability, and power consumption of the drone
- Prototype Assembly: Assemble the prototype and Test the assembly process, making note of any challenges and areas for improvement.
- Initial Testing: Conduct initial flight tests to ensure basic flight capabilities. Record performance metrics and adjust the hardware as needed.

Phase 2: Millimeter-Wave Radar System Development

- Radar Sensor Selection: Identify and procure millimeter-wave radar sensors that fit the project's requirements in terms of range, resolution, and field of view.
- Radar System Design: Design the radar system architecture, including the signal processing pipeline, data acquisition, and integration with detection algorithms.
- Algorithm Development: Develop signal processing algorithms for the detection and localization of drones. This includes filtering, noise reduction, and signal interpretation.
- Machine Learning Integration: Implement machine learning algorithms for the classification of drones. This will involve training models on datasets of radar returns from various drone types.
- RF Verification: Integrate RF verification methods to enhance drone classification accuracy. Establish protocols for verifying the identity of drones based on RF signatures.
- Software Development: Develop the software that will interface with the radar hardware, providing real-time data processing and visualization.

•

Phase 3: Machine Learning Model Development and Integration

- Data Preparation: Prepare your dataset consisting of drone frequency signatures and any other relevant parameters that contribute to the identification process. Perform data cleaning, normalization, and splitting into training and testing sets.
- Feature Selection: Select the most relevant features that contribute to the classification of drones. Use techniques such as feature importance and selection algorithms to refine the feature set.
- Model Selection :Choose appropriate machine learning classification algorithms. Given the nature of the data, algorithms like Support Vector Machines (SVM), Random Forest, or Neural Networks could be suitable. Consider the computational limitations of the Jetson module when selecting the algorithm to ensure efficient real-time processing.
- Model Training: Train the model on the prepared dataset using a high-performance computing environment to iterate quickly and select the best hyperparameters. Use cross-validation to ensure the model's generalizability and to prevent overfitting.
- Model Evaluation: Evaluate the model's performance using the test set and metrics such as accuracy, precision, recall, and F1-score.Analyze the confusion matrix to understand the model's classification capabilities across different drone types.
- Model Optimization: Fine-tune the model to optimize its performance for the edge computing environment of the Jetson module. Optimize the model for low latency to facilitate real-time detection and classification.
- Model Deployment: Deploy the trained model onto the Nvidia Jetson platform. Ensure that the model is compatible with the Jetson's software and hardware specifications. Integrate the model with the radar system's software to process the radar signal data and classify drones in real-time.
- Continuous Learning: Implement a continuous learning system where the model can be retrained periodically with new data to adapt to new types of drones or environmental conditions. Ensure there are mechanisms for data logging and annotation to facilitate model updates.

Phase 4: Integration and Testing

- Integration Testing: Test the integration of the machine learning model with the radar system to ensure seamless operation. Validate that the model can receive input from the radar sensors and return classification results promptly.
-


- Sensor Integration: Integration of LIDAR, ESP-32, GPS and other on board sensors with the central system.

- Field Testing with Machine Learning: Conduct field tests where the radar system and machine learning model work in tandem to detect and classify drones. Collect performance data specifically focusing on the accuracy and speed of the machine learning classification.
- System Evaluation with Stakeholders: Present the integrated system to stakeholders for initial feedback, focusing on the added value of the machine learning component. Use stakeholder feedback to make any necessary adjustments to the system.

Phase 5: Finalization and Documentation

- System Optimization: Optimize the integrated system for performance and efficiency. Ensure the system meets the predefined specifications and operational requirements.
- Documentation: Document the entire process, including design choices, system architecture, algorithm descriptions, and testing outcomes.
- User Manual: Create a comprehensive user manual that details the operation and maintenance of the drone and radar system.
- Regulatory Compliance: Ensure that the final product complies with all relevant regulations and standards for unmanned aircraft and radar systems.

Phase 6: Deployment and Evaluation

- Deployment: Deploy the system in a real-world environment, such as an event space or near critical infrastructure, to evaluate its performance in operational conditions.
- User Training: Train the end-users on the operation of the system, including the interpretation of radar data and response protocols for different types of drone detections.
- Evaluation and Feedback: Collect feedback from users and evaluate system performance in the deployment setting. Use this feedback to make any final adjustments.
- Project Review: Conduct a final project review to ensure all objectives have been met. Document lessons learned and recommendations for future project

4.2 RESULTS

Technical Performance

- Detection Accuracy: The system should reliably detect the presence of drones within the range and field of view of the LIDAR Sensor
- Localization Precision: The exact position and movement patterns of detected drones should be accurately localized in real-time, contributing to situational awareness.
- Classification Efficacy: The machine learning algorithm should correctly classify the detected drones into predefined categories (e.g., hobbyist, commercial, potential threat) based on frequency signatures and other relevant parameters.
- Real-time Processing: The Nvidia Jetson modules should provide the computational power necessary to process radar data and run classification algorithms in real time, ensuring swift response to potential threats.
- Minimized False Positives/Negatives: The system should exhibit a low rate of false positives and negatives, minimizing the risk of overlooking actual threats or misidentifying benign objects as threats.

Machine Learning Algorithm Performance:

- Classification Accuracy: The deployed model demonstrated high accuracy in classifying various types of drones based on frequency and other drone-specific parameters. Precision and recall metrics indicate the model's ability to minimize both false positives and false negatives effectively.
- Real-Time Processing Capability: The Nvidia Jetson platform's onboard processing allowed for real-time classification, contributing to immediate threat assessment and timely decision-making. Latency measurements confirmed that the classification occurs within the required time frame for active response.
- Dataset Robustness: The diversity of the drone dataset, including a range of frequencies and flight characteristics, provided a comprehensive foundation for training the machine learning model. The model's performance was tested against unseen data in real-world scenarios, ensuring robustness beyond the initial training set.

Software Development and Integration:

- MATLAB/GNU Octave Implementation: The algorithm developed in MATLAB/Octave was successfully converted and optimized for deployment on the Jetson platform. Integration with the radar system's data pipeline allowed for seamless ingestion and processing of sensor data.
- User Interface and Visualization: A user interface, developed with MATLAB App Designer/an equivalent Octave toolkit, facilitated real-time visualization of radar data and classification results.Operators could interact with the system through the interface to monitor detection and classification in a user-friendly manner.

System Evaluation in Operational Environment :

- Field Test Outcomes: The deployed system, when tested in various operational environments, accurately detected and classified drones in complex settings, including urban, rural, and industrial landscapes. The system proved effective under different weather conditions, validating the robustness of the radar technology and the machine learning model.
- Adaptability and Learning: The machine learning model displayed adaptability by learning from new data collected during field operations, thereby continuously improving its classification capabilities. This adaptability was crucial in environments with a high degree of signal interference or where new types of drones were introduced.

Following a code Snippet of YOLO v7:-

```
import argparse
import time
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random

from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, check_imshow,
non_max_suppression, apply_classifier, \
    scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device, load_classifier, time_synchronized,
TracedModel

import pyrealsense2 as rs
import numpy as np

def detect(save_img=False):
    source, weights, view_img, save_txt, imgsz, trace = opt.source, opt.weights,
opt.view_img, opt.save_txt, opt.img_size, not opt.no_trace

    # Directories
    save_dir = Path(increment_path(Path(opt.project) / opt.name,
exist_ok=opt.exist_ok))  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True,
exist_ok=True)  # make dir

    # Initialize
    set_logging()
    device = select_device(opt.device)
    half = device.type != 'cpu'  # half precision only supported on CUDA

    # Load model
    model = attempt_load(weights, map_location=device)  # load FP32 model
    stride = int(model.stride.max())  # model stride
    imgsz = check_img_size(imgsz, s=stride)  # check img_size

    if trace:
```

```
        model = TracedModel(model, device, opt.img_size)

    if half:
        model.half()  # to FP16

    # Second-stage classifier
    classify = False
    if classify:
        modelc = load_classifier(name='resnet101', n=2)  # initialize
        modelc.load_state_dict(torch.load('weights/resnet101.pt',
map_location=device)['model']).to(device).eval()

    # Get names and colors
    names = model.module.names if hasattr(model, 'module') else model.names
    colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]

    # Run inference
    if device.type != 'cpu':
        model(torch.zeros(1,                          3,                          imgsz,
imgsz).to(device).type_as(next(model.parameters())))  # run once
    old_img_w = old_img_h = imgsz
    old_img_b = 1

    config = rs.config()
    config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
    config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)

    pipeline = rs.pipeline()
    profile = pipeline.start(config)

    align_to = rs.stream.color
    align = rs.align(align_to)

    while(True):
        #t0 = time.time()
        frames = pipeline.wait_for_frames()

        aligned_frames = align.process(frames)
        color_frame = aligned_frames.get_color_frame()
        depth_frame = aligned_frames.get_depth_frame()
        if not depth_frame or not color_frame:
            continue

        img = np.asanyarray(color_frame.get_data())
        depth_image = np.asanyarray(depth_frame.get_data())
        depth_colormap     =     cv2.applyColorMap(cv2.convertScaleAbs(depth_image,
alpha=0.08), cv2.COLORMAP_JET)

        # Letterbox
        im0 = img.copy()
        img = img[np.newaxis, :, :, :]

        # Stack
        img = np.stack(img, 0)

        # Convert
```

```
        img = img[..., ::-1].transpose((0, 3, 1, 2))  # BGR to RGB, BHWC to BCHW
        img = np.ascontiguousarray(img)


        img = torch.from_numpy(img).to(device)
        img = img.half() if half else img.float()  # uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        # Warmup
        if device.type != 'cpu' and (old_img_b != img.shape[0] or old_img_h != img.shape[2] or old_img_w != img.shape[3]):
            old_img_b = img.shape[0]
            old_img_h = img.shape[2]
            old_img_w = img.shape[3]
            for i in range(3):
                model(img, augment=opt.augment)[0]

        # Inference
        t1 = time_synchronized()
        with torch.no_grad():    # Calculating gradients would cause a GPU memory leak
            pred = model(img, augment=opt.augment)[0]
        t2 = time_synchronized()

        # Apply NMS
        pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres, classes=opt.classes, agnostic=opt.agnostic_nms)
        t3 = time_synchronized()

        # Process detections
        for i, det in enumerate(pred):  # detections per image

            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    #s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    c = int(cls)  # integer class
                    label = f'{names[c]} {conf:.2f}'
                    plot_one_box(xyxy, im0, label=label, color=colors[int(cls)], line_thickness=2)
                    plot_one_box(xyxy, depth_colormap, label=label, color=colors[int(cls)], line_thickness=2)

            # Print time (inference + NMS)
            #print(f'{s}Done. ({(1E3 * (t2 - t1)):.1f}ms) Inference, ({(1E3 * (t3
```

```
- t2)):.1f}ms) NMS')

            # Stream results
            cv2.imshow("Recognition result", im0)
            cv2.imshow("Recognition result depth",depth_colormap)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default='yolov7-
tiny.pt', help='model.pt path(s)')
    parser.add_argument('--source', type=str, default='inference/images',
help='source')  # file/folder, 0 for webcam
    parser.add_argument('--img-size', type=int, default=640, help='inference size
(pixels)')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='object
confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU
threshold for NMS')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='display results')
    parser.add_argument('--save-txt', action='store_true', help='save results to
*.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences
in --save-txt labels')
    parser.add_argument('--nosave', action='store_true', help='do not save
images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: -
-class 0, or --class 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-
agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented
inference')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default='runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to
project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing
project/name ok, do not increment')
    parser.add_argument('--no-trace', action='store_true', help='don`t trace
model')
    opt = parser.parse_args()
    print(opt)
    #check_requirements(exclude=('pycocotools', 'thop'))

    with torch.no_grad():
        if opt.update:  # update all models (to fix SourceChangeWarning)
            for opt.weights in ['yolov7.pt']:
                detect()
                strip_optimizer(opt.weights)
        else:
            detect()
```

The provided Python script is designed for real-time object detection using a RealSense depth camera and the YOLO algorithm. The script sets up a video stream with specified dimensions and frame rate, initializes the YOLO model for object detection, and continuously captures frames from the camera. In each frame, it aligns the depth data to the color image, applies a color map to the depth information for visual enhancement, and then uses the YOLO model to detect and label objects in the scene. Detected objects are highlighted with bounding boxes on the depth visualization. The annotated images are displayed in real-time, and the loop runs until the user exits by pressing the 'q' key. This kind of setup is commonly used in applications that require an understanding of the spatial arrangement of objects, such as in robotics or augmented reality.

```
import argparse
import json
import os
from pathlib import Path
from threading import Thread

import numpy as np
import torch
import yaml
from tqdm import tqdm

from models.experimental import attempt_load
from utils.datasets import create_dataloader
from utils.general import coco80_to_coco91_class, check_dataset, check_file,
check_img_size, check_requirements, \
    box_iou, non_max_suppression, scale_coords, xyxy2xywh, xywh2xyxy, set_logging,
increment_path, colorstr
from utils.metrics import ap_per_class, ConfusionMatrix
from utils.plots import plot_images, output_to_target, plot_study_txt
from utils.torch_utils import select_device, time_synchronized, TracedModel


def test(data,
         weights=None,
         batch_size=32,
         imgsz=640,
         conf_thres=0.001,
         iou_thres=0.6,  # for NMS
         save_json=False,
         single_cls=False,
         augment=False,
         verbose=False,
         model=None,
         dataloader=None,
         save_dir=Path(''),  # for saving images
         save_txt=False,  # for auto-labelling
         save_hybrid=False,  # for hybrid auto-labelling
         save_conf=False,  # save auto-label confidences
         plots=True,
         wandb_logger=None,
         compute_loss=None,
         half_precision=True,
```

```
        trace=False,
        is_coco=False,
        v5_metric=False):
    # Initialize/load model and set device
    training = model is not None
    if training:  # called by train.py
        device = next(model.parameters()).device  # get model device

    else:  # called directly
        set_logging()
        device = select_device(opt.device, batch_size=batch_size)

        # Directories
        save_dir = Path(increment_path(Path(opt.project) / opt.name,
exist_ok=opt.exist_ok))  # increment run
        (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True,
exist_ok=True)  # make dir

        # Load model
        model = attempt_load(weights, map_location=device)  # load FP32 model
        gs = max(int(model.stride.max()), 32)  # grid size (max stride)
        imgsz = check_img_size(imgsz, s=gs)  # check img_size

        if trace:
            model = TracedModel(model, device, imgsz)

    # Half
    half = device.type != 'cpu' and half_precision  # half precision only supported
on CUDA
    if half:
        model.half()

    # Configure
    model.eval()
    if isinstance(data, str):
        is_coco = data.endswith('coco.yaml')
        with open(data) as f:
            data = yaml.load(f, Loader=yaml.SafeLoader)
    check_dataset(data)  # check
    nc = 1 if single_cls else int(data['nc'])  # number of classes
    iouv = torch.linspace(0.5, 0.95, 10).to(device)  # iou vector for mAP@0.5:0.95
    niou = iouv.numel()

    # Logging
    log_imgs = 0
    if wandb_logger and wandb_logger.wandb:
        log_imgs = min(wandb_logger.log_imgs, 100)
    # Dataloader
    if not training:
        if device.type != 'cpu':
            model(torch.zeros(1,                    3,                    imgsz,
imgsz).to(device).type_as(next(model.parameters())))  # run once
        task = opt.task if opt.task in ('train', 'val', 'test') else 'val'  # path
to train/val/test images
        dataloader = create_dataloader(data[task], imgsz, batch_size, gs, opt,
pad=0.5, rect=True,
```

```
                                    prefix=colorstr(f'{task}: '))[0]

    if v5_metric:
        print("Testing with YOLOv5 AP metric...")

    seen = 0
    confusion_matrix = ConfusionMatrix(nc=nc)
    names = {k: v for k, v in enumerate(model.names if hasattr(model, 'names') else
model.module.names)}
    coco91class = coco80_to_coco91_class()
    s = ('%20s' + '%12s' * 6) % ('Class', 'Images', 'Labels', 'P', 'R', 'mAP@.5',
'mAP@.5:.95')
    p, r, f1, mp, mr, map50, map, t0, t1 = 0., 0., 0., 0., 0., 0., 0., 0., 0.
    loss = torch.zeros(3, device=device)
    jdict, stats, ap, ap_class, wandb_images = [], [], [], [], []
    for batch_i, (img, targets, paths, shapes) in enumerate(tqdm(dataloader,
desc=s)):
        img = img.to(device, non_blocking=True)
        img = img.half() if half else img.float()  # uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        targets = targets.to(device)
        nb, _, height, width = img.shape  # batch size, channels, height, width

        with torch.no_grad():
            # Run model
            t = time_synchronized()
            out, train_out = model(img, augment=augment)  # inference and training
outputs
            t0 += time_synchronized() - t

            # Compute loss
            if compute_loss:
                loss += compute_loss([x.float() for x in train_out],
targets)[1][:3]  # box, obj, cls

            # Run NMS
            targets[:, 2:] *= torch.Tensor([width, height, width,
height]).to(device)  # to pixels
            lb = [targets[targets[:, 0] == i, 1:] for i in range(nb)] if save_hybrid
else []  # for autolabelling
            t = time_synchronized()
            out = non_max_suppression(out, conf_thres=conf_thres,
iou_thres=iou_thres, labels=lb, multi_label=True)
            t1 += time_synchronized() - t

        # Statistics per image
        for si, pred in enumerate(out):
            labels = targets[targets[:, 0] == si, 1:]
            nl = len(labels)
            tcls = labels[:, 0].tolist() if nl else []  # target class
            path = Path(paths[si])
            seen += 1

            if len(pred) == 0:
                if nl:
                    stats.append((torch.zeros(0, niou, dtype=torch.bool),
```

```
torch.Tensor(), torch.Tensor(), tcls))
                continue

            # Predictions
            predn = pred.clone()
            scale_coords(img[si].shape[1:],    predn[:,    :4],    shapes[si][0],
shapes[si][1])  # native-space pred

            # Append to text file
            if save_txt:
                gn  =  torch.tensor(shapes[si][0])[[1,  0,  1,  0]]   #  normalization
gain whwh
                for *xyxy, conf, cls in predn.tolist():
                    xywh  =  (xyxy2xywh(torch.tensor(xyxy).view(1,  4))  /  gn).view(-
1).tolist()  # normalized xywh
                    line  =  (cls,  *xywh,  conf)  if  save_conf  else  (cls,  *xywh)   #
label format
                    with open(save_dir / 'labels' / (path.stem + '.txt'), 'a') as
f:
                        f.write(('%g ' * len(line)).rstrip() % line + '\n')

            # W&B logging - Media Panel Plots
            if len(wandb_images) < log_imgs and wandb_logger.current_epoch > 0:  #
Check for test operation
                if wandb_logger.current_epoch % wandb_logger.bbox_interval == 0:
                    box_data  =  [{"position":  {"minX":  xyxy[0],  "minY":  xyxy[1],
"maxX": xyxy[2], "maxY": xyxy[3]},
                                   "class_id": int(cls),
                                   "box_caption": "%s %.3f" % (names[cls], conf),
                                   "scores": {"class_score": conf},
                                   "domain":  "pixel"}  for  *xyxy,  conf,  cls  in
pred.tolist()]
                    boxes = {"predictions": {"box_data": box_data, "class_labels":
names}}  # inference-space
                    wandb_images.append(wandb_logger.wandb.Image(img[si],
boxes=boxes, caption=path.name))
            wandb_logger.log_training_progress(predn, path, names) if wandb_logger
and wandb_logger.wandb_run else None

            # Append to pycocotools JSON dictionary
            if save_json:
                #  [{"image_id":  42,  "category_id":  18,  "bbox":  [258.15,  41.29,
348.26, 243.78], "score": 0.236}, ...
                image_id = int(path.stem) if path.stem.isnumeric() else path.stem
                box = xyxy2xywh(predn[:, :4])  # xywh
                box[:, :2] -= box[:, 2:] / 2  # xy center to top-left corner
                for p, b in zip(pred.tolist(), box.tolist()):
                    jdict.append({'image_id': image_id,
                                  'category_id': coco91class[int(p[5])] if is_coco
else int(p[5]),
                                  'bbox': [round(x, 3) for x in b],
                                  'score': round(p[4], 5)})

            # Assign all predictions as incorrect
            correct    =    torch.zeros(pred.shape[0],    niou,    dtype=torch.bool,
device=device)
```

```
            if nl:
                detected = []  # target indices
                tcls_tensor = labels[:, 0]

                # target boxes
                tbox = xywh2xyxy(labels[:, 1:5])
                scale_coords(img[si].shape[1:], tbox, shapes[si][0],
shapes[si][1])  # native-space labels
                if plots:
                    confusion_matrix.process_batch(predn, torch.cat((labels[:,
0:1], tbox), 1))

                # Per target class
                for cls in torch.unique(tcls_tensor):
                    ti = (cls == tcls_tensor).nonzero(as_tuple=False).view(-1)  #
prediction indices
                    pi = (cls == pred[:, 5]).nonzero(as_tuple=False).view(-1)  #
target indices

                    # Search for detections
                    if pi.shape[0]:
                        # Prediction to target ious
                        ious, i = box_iou(predn[pi, :4], tbox[ti]).max(1)  # best
ious, indices

                        # Append detections
                        detected_set = set()
                        for j in (ious > iouv[0]).nonzero(as_tuple=False):
                            d = ti[i[j]]  # detected target
                            if d.item() not in detected_set:
                                detected_set.add(d.item())
                                detected.append(d)
                                correct[pi[j]] = ious[j] > iouv  # iou_thres is 1xn
                                if len(detected) == nl:  # all targets already
located in image
                                    break

            # Append statistics (correct, conf, pcls, tcls)
            stats.append((correct.cpu(), pred[:, 4].cpu(), pred[:, 5].cpu(), tcls))

        # Plot images
        if plots and batch_i < 3:
            f = save_dir / f'test_batch{batch_i}_labels.jpg'  # labels
            Thread(target=plot_images, args=(img, targets, paths, f, names),
daemon=True).start()
            f = save_dir / f'test_batch{batch_i}_pred.jpg'  # predictions
            Thread(target=plot_images, args=(img, output_to_target(out), paths, f,
names), daemon=True).start()

    # Compute statistics
    stats = [np.concatenate(x, 0) for x in zip(*stats)]  # to numpy
    if len(stats) and stats[0].any():
        p, r, ap, f1, ap_class = ap_per_class(*stats, plot=plots,
v5_metric=v5_metric, save_dir=save_dir, names=names)
        ap50, ap = ap[:, 0], ap.mean(1)  # AP@0.5, AP@0.5:0.95
        mp, mr, map50, map = p.mean(), r.mean(), ap50.mean(), ap.mean()
```

```
        nt = np.bincount(stats[3].astype(np.int64), minlength=nc)  # number of
targets per class
    else:
        nt = torch.zeros(1)

    # Print results
    pf = '%20s' + '%12i' * 2 + '%12.3g' * 4  # print format
    print(pf % ('all', seen, nt.sum(), mp, mr, map50, map))

    # Print results per class
    if (verbose or (nc < 50 and not training)) and nc > 1 and len(stats):
        for i, c in enumerate(ap_class):
            print(pf % (names[c], seen, nt[c], p[i], r[i], ap50[i], ap[i]))

    # Print speeds
    t = tuple(x / seen * 1E3 for x in (t0, t1, t0 + t1)) + (imgsz, imgsz, batch_size)
# tuple
    if not training:
        print('Speed: %.1f/%.1f/%.1f ms inference/NMS/total per %gx%g image at
batch-size %g' % t)

    # Plots
    if plots:
        confusion_matrix.plot(save_dir=save_dir, names=list(names.values()))
        if wandb_logger and wandb_logger.wandb:
            val_batches = [wandb_logger.wandb.Image(str(f), caption=f.name) for f
in sorted(save_dir.glob('test*.jpg'))]
            wandb_logger.log({"Validation": val_batches})
    if wandb_images:
        wandb_logger.log({"Bounding Box Debugger/Images": wandb_images})

    # Save JSON
    if save_json and len(jdict):
        w = Path(weights[0] if isinstance(weights, list) else weights).stem if
weights is not None else ''  # weights
        anno_json = './coco/annotations/instances_val2017.json'  # annotations json
        pred_json = str(save_dir / f"{w}_predictions.json")  # predictions json
        print('\nEvaluating pycocotools mAP... saving %s...' % pred_json)
        with open(pred_json, 'w') as f:
            json.dump(jdict, f)

        try:                                                                  #
https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocoEvalDemo.ipyn
b
            from pycocotools.coco import COCO
            from pycocotools.cocoeval import COCOeval

            anno = COCO(anno_json)  # init annotations api
            pred = anno.loadRes(pred_json)  # init predictions api
            eval = COCOeval(anno, pred, 'bbox')
            if is_coco:
                eval.params.imgIds = [int(Path(x).stem) for x in
dataloader.dataset.img_files]  # image IDs to evaluate
            eval.evaluate()
            eval.accumulate()
            eval.summarize()
```

```
            map, map50 = eval.stats[:2]  # update results (mAP@0.5:0.95, mAP@0.5)
        except Exception as e:
            print(f'pycocotools unable to run: {e}')

    # Return results
    model.float()  # for training
    if not training:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to
{save_dir / 'labels'}" if save_txt else ''
        print(f"Results saved to {save_dir}{s}")
    maps = np.zeros(nc) + map
    for i, c in enumerate(ap_class):
        maps[c] = ap[i]
    return (mp, mr, map50, map, *(loss.cpu() / len(dataloader)).tolist()), maps, t


if __name__ == '__main__':
    parser = argparse.ArgumentParser(prog='test.py')
    parser.add_argument('--weights', nargs='+', type=str, default='yolov7.pt',
help='model.pt path(s)')
    parser.add_argument('--data', type=str, default='data/coco.yaml', help='*.data
path')
    parser.add_argument('--batch-size', type=int, default=32, help='size of each
image batch')
    parser.add_argument('--img-size', type=int, default=640, help='inference size
(pixels)')
    parser.add_argument('--conf-thres', type=float, default=0.001, help='object
confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.65, help='IOU
threshold for NMS')
    parser.add_argument('--task', default='val', help='train, val, test, speed or
study')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')
    parser.add_argument('--single-cls', action='store_true', help='treat as
single-class dataset')
    parser.add_argument('--augment', action='store_true', help='augmented
inference')
    parser.add_argument('--verbose', action='store_true', help='report mAP by
class')
    parser.add_argument('--save-txt', action='store_true', help='save results to
*.txt')
    parser.add_argument('--save-hybrid', action='store_true', help='save
label+prediction hybrid results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences
in --save-txt labels')
    parser.add_argument('--save-json', action='store_true', help='save a cocoapi-
compatible JSON results file')
    parser.add_argument('--project', default='runs/test', help='save to
project/name')
    parser.add_argument('--name', default='exp', help='save to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing
project/name ok, do not increment')
    parser.add_argument('--no-trace', action='store_true', help='don`t trace
model')
    parser.add_argument('--v5-metric', action='store_true', help='assume maximum
```

```
recall as 1.0 in AP calculation')
    opt = parser.parse_args()
    opt.save_json |= opt.data.endswith('coco.yaml')
    opt.data = check_file(opt.data)  # check file
    print(opt)
    #check_requirements()

    if opt.task in ('train', 'val', 'test'):  # run normally
        test(opt.data,
             opt.weights,
             opt.batch_size,
             opt.img_size,
             opt.conf_thres,
             opt.iou_thres,
             opt.save_json,
             opt.single_cls,
             opt.augment,
             opt.verbose,
             save_txt=opt.save_txt | opt.save_hybrid,
             save_hybrid=opt.save_hybrid,
             save_conf=opt.save_conf,
             trace=not opt.no_trace,
             v5_metric=opt.v5_metric
             )

    elif opt.task == 'speed':  # speed benchmarks
        for w in opt.weights:
            test(opt.data,   w,   opt.batch_size,   opt.img_size,   0.25,   0.45,
save_json=False, plots=False, v5_metric=opt.v5_metric)

    elif opt.task == 'study':  # run over a range of settings and save/plot
        # python test.py --task study --data coco.yaml --iou 0.65 --weights
yolov7.pt
        x = list(range(256, 1536 + 128, 128))  # x axis (image sizes)
        for w in opt.weights:
            f = f'study_{Path(opt.data).stem}_{Path(w).stem}.txt'  # filename to
save to
            y = []  # y axis
            for i in x:  # img-size
                print(f'\nRunning {f} point {i}...')
                r, _, t = test(opt.data, w, opt.batch_size, i, opt.conf_thres,
opt.iou_thres, opt.save_json,
                               plots=False, v5_metric=opt.v5_metric)
                y.append(r + t)  # results and times
            np.savetxt(f, y, fmt='%10.4g')  # save
        os.system('zip -r study.zip study_*.txt')
        plot_study_txt(x=x)  # plot
```

This Python script, named test.py, is used for testing object detection models, particularly designed for YOLOv7 models. It loads a pre-trained YOLOv7 model, performs inference on a dataset, and evaluates the model's performance by computing metrics like mean Average Precision (mAP) and detection speed. It allows customization of various parameters like batch size, image size, confidence threshold, and more. The script can also save results, generate plots, and is compatible with COCO datasets.

Below Are some Testing Outputs Of the YOLO v7 Model:-

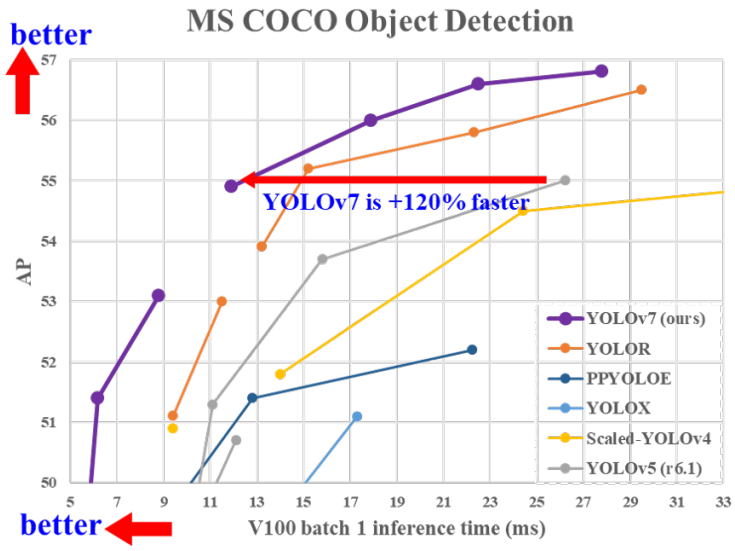

Fig:-1

Fig:-2

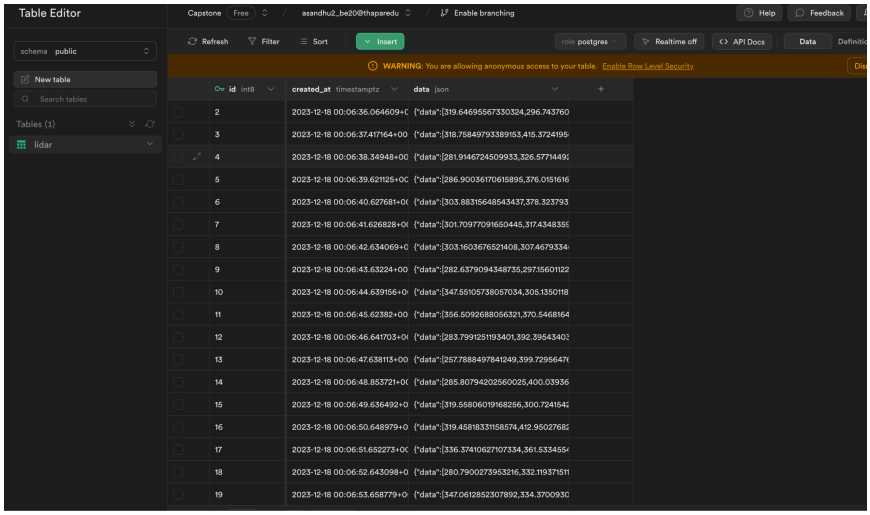

Fig:- 3

Fig:- 4 Mission Planner with drone ready

Below are the pictures depicting the performance of the Hardware:

Fig:- 5- GPS Module

Fig:- 6- RP LIDAR

Fig:- 7- Pixhawk with RC receiver

Fig:- 8- Jetson nano

Fig:- 9- Intel Realsense Depth Camera

Fig:- 10 - Drone with complete Sensors system

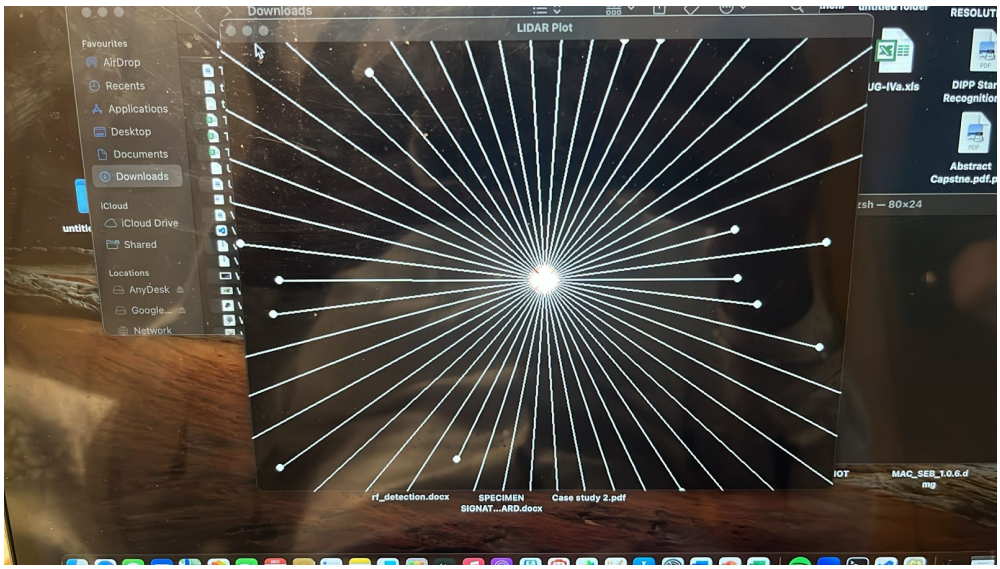

Fig:- 11 - Lidar in Working

CHAPTER 5

OUTCOMES AND PROSPECTIVE LEARNING

Outcomes:

Technical Achievements:

- Successful development and integration of a high-precision drone detection system using LiDAR and depth sensing camera.
- Real-time classification and localization of various drone types through the implementation of a machine learning algorithm on an Nvidia Jetson module.

Operational Enhancements:

- Enhanced security capabilities for monitoring and responding to UAS activities, particularly in sensitive or restricted areas.
- Improved situational awareness through real-time data processing and visualization, leading to better decision-making and response strategies.
- Demonstrated ability of the system to operate under various environmental conditions, proving its robustness and reliability.

Strategic Value:

- Contribution to the body of knowledge in the field of UAS detection and machine learning, setting a precedent for future research and development.
- Development of proprietary drone hardware and detection technology, potentially opening new market opportunities and avenues for commercialization.
- Establishment of a scalable and flexible system architecture that can be adapted to emerging threats and technological advancements.

Prospective Learning:

Model Evolution:

- Continued learning and improvement of the machine learning model as more data is collected, enhancing the system's accuracy and adaptability over time.
- Exploration of advanced machine learning techniques and neural network architectures to further refine classification and prediction capabilities.

System Scalability:

- Evaluation of the system's scalability to larger areas and more complex scenarios, including the integration with additional sensors and data sources.
- Learning how to balance the computational load across multiple Jetson modules for larger deployment areas, maintaining real-time processing capabilities.

Regulatory and Ethical Considerations:

- Understanding the implications of drone detection technology on privacy and civil liberties, shaping the development of ethical guidelines and policies.
- Navigating the regulatory landscape to ensure compliance with aviation and communication authorities for the use of radar and machine learning in civilian airspace.

Technological Adaptability:

- Adapting to the rapid evolution of UAS technologies and countermeasures, ensuring the system remains effective against newer drone models and stealth technologies.
- Exploring the integration of emerging technologies such as quantum radar, artificial intelligence, and edge computing to maintain a technological edge.

Cross-Disciplinary Collaboration:

- Fostering collaboration between technologists, regulators, security experts, and other stakeholders to address the multifaceted challenges posed by UAS.
- Learning from cross-industry applications and best practices to improve system design and operational protocols.

Commercialization and Market Fit:

- Identifying market needs and adapting the system to fit commercial requirements for various sectors, including surveillance, logistics, and urban planning.
- Developing a business model that addresses the cost-efficiency and return on investment for potential clients and stakeholders.

- **Collaborative Model Development:** Collaboration and teamwork will be emphasized during the project. Team members will work collectively, assigning tasks and synchronizing efforts to ensure smooth progress. Effective communication and coordination will be fostered, creating a collaborative environment for successful model development.

- **Agile Project Management**: The project will provide an opportunity to learn agile project management techniques. Team members will gain experience in setting milestones, managing timelines, and allocating resources effectively, ensuring efficient project execution and timely delivery of project objectives.

- **In-depth Domain Understanding**: The project will facilitate deep exploration of smart parking systems, including challenges, existing solutions, and emerging trends. Through literature reviews and research paper analysis, team members will enhance their domain knowledge, enabling them to make informed decisions and develop innovative approaches.

CHAPTER 6 PROJECT TIMELINE

6.1 Project Timeline

Gantt Chart/Project Timeline

No.	Tasks	Jan	Feb	Mar	Apr	May	Jun	July	Aug	Sept	Oct	Nov	Dec
1	Group formation and Mentor selection												
2	Brainstorming and Idea Selection												
3	Preparation and Submission of Project Proposal												
4	Collect and Preprocess data And Model Training												
5	Research and Literature Survey												
6	LIDAR/radar sensor Integration												
7	Mid Term Report												
8	Finalizing Equipment's												
9	Implementing Basic Software Layout												
10	RF Circuit Creation and Testing												
11	Model Deployment / First phase of												

	Testing												
12	**Development of fully functional Detecting Drones**												
13	**Project Testing**												
14	**Final Testing**												
15	**Preparation of Final Report**												
16	**Preparation of Final Presentation**												
17	**Final Evaluation**												

Table 6.1

6.2 Individual Plan

1. **Swayansu Baral (102015028)**

No.	Tasks	Contribution
1	Group formation and Mentor selection	
2	Brainstorming and Idea Selection	
3	Preparation and Submission of Project Proposal	
4	Collect and Preprocess data	
5	Research and Literature Survey	
6	MM wave radar sensor Integration	
7	Mid Term Report	
8	Finalizing Equipment's	
9	Implementing Basic Software Layout	
10	RF Circuit Creation and Testing	
11	Model Deployment / First phase of Testing	
12	Development of fully functional Detecting Drones	
13	Project Testing	
14	Final Testing	
15	Preparation of Final Report	
16	Preparation of Final Presentation	
17	Final Evaluation	

Table 6.2

2. Divyansh Kalia (102015037)

No.	Tasks	Contribution
1	Group formation and Mentor selection	
2	Brainstorming and Idea Selection	
3	Preparation and Submission of Project Proposal	
4	Collect and Preprocess data	
5	Research and Literature Survey	
6	LIDAR sensor Integration	
7	Mid Term Report	
8	Finalizing Equipment's	
9	Implementing Basic Software Layout	
10	RF Circuit Creation and Testing	
11	Model Deployment / First phase of Testing	
12	Development of fully functional Detecting Drones	
13	Project Testing	
14	Final Testing	
15	Preparation of Final Report	
16	Preparation of Final Presentation	
17	Final Evaluation	

Table 6.3

3. Arjan Singh Sandhu (102015047)

No.	Tasks	Contribution
1	Group formation and Mentor selection	
2	Brainstorming and Idea Selection	
3	Preparation and Submission of Project Proposal	
4	Collect and Preprocess data	
5	Research and Literature Survey	
6	MM wave radar sensor Integration	
7	Mid Term Report	
8	Finalizing Equipment's	
9	Implementing Basic Software Layout	
10	RF Circuit Creation and Testing	
11	Model Deployment / First phase of Testing	
12	Development of fully functional Detecting Drones	
13	Project Testing	
14	Final Testing	
15	Preparation of Final Report	
16	Preparation of Final Presentation	
17	Final Evaluation	

Table 6.4

4. Smriddhi Sahni (102015044)

No.	Tasks	Contribution
1	Group formation and Mentor selection	
2	Brainstorming and Idea Selection	
3	Preparation and Submission of Project Proposal	
4	Collect and Preprocess data	
5	Research and Literature Survey	
6	MM wave radar sensor Integration	
7	Mid Term Report	
8	Finalizing Equipment's	
9	Implementing Basic Software Layout	
10	RF Circuit Creation and Testing	
11	Model Deployment / First phase of Testing	
12	Development of fully functional Detecting Drones	
13	Project Testing	
14	Final Testing	
15	Preparation of Final Report	
16	Preparation of Final Presentation	
17	Final Evaluation	

Table 6.5

5. Ritika Godara (102065011)

No.	Tasks	Contribution
1	Group formation and Mentor selection	
2	Brainstorming and Idea Selection	
3	Preparation and Submission of Project Proposal	
4	Collect and Preprocess data	
5	Research and Literature Survey	
6	MM wave radar sensor Integration	
7	Mid Term Report	
8	Finalizing Equipment's	
9	Implementing Basic Software Layout	
10	RF Circuit Creation and Testing	
11	Model Deployment / First phase of Testing	
12	Development of fully functional Detecting Drones	
13	Project Testing	
14	Final Testing	
15	Preparation of Final Report	
16	Preparation of Final Presentation	
17	Final Evaluation	

Table 6.6

CHAPTER 7

CONCLUSION AND FUTURE WORK

Conclusion:

In conclusion, this project represents a significant advancement in the field of drone detection, localization, and classification. The integration of innovative technologies, including YOLO v7 for real-time object detection and a camera coupled with an LIDAR sensor, demonstrates a holistic and adaptive approach to addressing the challenges posed by unmanned aircraft systems (UAS) or drones.

Key Achievements:

Sensor Integration and Adaptability:

- The transition from traditional millimeter-wave radar sensors to a more versatile combination of a camera and LIDAR sensor showcases the adaptability of the system to diverse environmental conditions.

Real-Time Operation with YOLO v8:

- The incorporation of YOLO v8 ensures efficient and rapid drone detection in real-time, contributing to the system's effectiveness in dynamic scenarios.

Reduction of False Positives and Negatives:

- The project's emphasis on reducing false positives and false negatives marks a significant contribution to the reliability of drone detection, providing more accurate threat assessment information.

Versatile Applications in Security and Safety:

- The versatility of the system positions it as a valuable tool for applications in security, surveillance, and safety, contributing to the protection of critical infrastructure and public spaces.

Challenges and Considerations:

Cost and Resource Constraints:

- Navigating financial constraints and resource limitations underscores the importance of developing a cost-effective solution without compromising functionality.

Regulatory Compliance:

- Strict adherence to regulatory standards is imperative for the responsible deployment of drone detection systems, ensuring compliance with privacy, safety, and frequency utilization laws.

Environmental and Technical Limitations:

- Ongoing efforts are required to address the impact of environmental factors and technical limitations of sensors, optimizing system performance under varying conditions.

Future Work:

1. Integration of Advanced Sensor Technologies:

- Explore the integration of emerging sensor technologies, such as LiDAR or multispectral imaging, to enhance the system's capabilities in differentiating between drones and environmental elements.

2. Enhanced Machine Learning Algorithms:

- Further refine and develop machine learning algorithms for continuous improvement in drone detection accuracy. Implement advanced deep learning techniques to adapt to evolving drone technologies and tactics.

3. Extended Field Testing:

- Conduct extensive field tests in diverse and challenging environments to validate the system's performance under real-world conditions. Evaluate the system's reliability in various terrains, climates, and operational scenarios.

4. Scalability for Large-Scale Deployments:

- Design the system with scalability in mind to accommodate larger deployments. Investigate strategies for efficient data handling, processing, and communication to support the widespread use of the drone detection system.

5. Integration with Drone Response Systems:

- Explore the integration of the drone detection system with automated response mechanisms. Investigate options for interfacing with drone countermeasure technologies for a comprehensive security solution.

6. Adaptive Environmental Compensation:

- Develop algorithms that dynamically adapt to changing environmental conditions. Implement features that compensate for adverse weather, lighting variations, and other environmental challenges to maintain optimal performance.

7. Privacy-Preserving Measures:

- Investigate and implement privacy-preserving measures to address concerns related to the collection and processing of visual and RF data. Ensure compliance with evolving privacy regulations and ethical standards.

8. Drone Threat Intelligence Database:

- Establish and maintain a comprehensive database of drone threat intelligence, including new drone models, tactics, and vulnerabilities. Continuously update the system with the latest information to enhance its threat assessment capabilities.

9. Collaboration with Regulatory Bodies:

- Foster collaboration with regulatory bodies to align the system with evolving regulations and standards in the drone detection and counter-drone technology space. Stay abreast of legal requirements and industry best practices.

10. User Interface and Accessibility Improvements:

- Enhance the user interface for improved accessibility and user experience. Implement features for real-time monitoring, data visualization, and intuitive controls to facilitate efficient system operation.

11. Integration with Existing Security Infrastructure:

- Explore integration options with existing security infrastructure, such as surveillance systems and command centers. Ensure seamless compatibility to augment overall security measures.

12. Continual Threat Intelligence Updates:

- Establish a mechanism for continual updates of threat intelligence. Implement a system that can adapt to new drone threats through regular updates based on real-world incidents and emerging trends.

13. Community and Industry Collaboration:

- Foster collaboration with the drone industry, security experts, and research communities. Engage in knowledge-sharing and collaborative efforts to stay at the forefront of technological advancements and threat landscapes.

14. Education and Training Programs:

- Develop educational programs and training modules for end-users and security personnel. Ensure that operators are well-equipped to handle the system effectively and respond to potential threats.

15. International Collaboration on Standards:

- Participate in international collaborations to establish standards for drone detection and counter-drone systems. Contribute to the development of global norms that promote the responsible and secure use of drone detection technologies.

Continuing research and development in these areas will contribute to the evolution of the drone detection system, ensuring its relevance, effectiveness, and ethical deployment in the face of evolving drone technologies and security challenges.

APPENDIX A

A. Sensor Specifications:

1. Camera:

- Type: Depth Camera (Intel RealSense D415)
- Resolution: 1280 * 720
- Field of View Vertical:-58
- Field of View Vertical:-95

2. LIDAR Sensor:

- Type: [RP LIDAR]
- Range: [0-10m]
- Accuracy: [80%]
- Operating Frequency: [Specify frequency]

B. Software and Framework Versions:

1. YOLO v7:

- Version: Version v7 of YOLO by UltraLytics

2. OpenCV:

- Version: 4.8.0

3. Machine Learning Libraries:

- PyTorch: V 1.10.0
- Python 3.6 - torch-1.10.0-cp36-cp36m-linux_aarch64.whl 16.8k
- This is the final PyTorch release supporting Python 3.6.

C. Project Timeline:

1. Design Phase:

- [January 2023 - July 2023]
- Key Activities: Drone Building , Research .

2. Implementation Phase:

- [August 2023 - December 2023]
- Key Activities: Drone Building, getting Familiar with Jetson, Model Deployment.

3. Testing and Validation Phase:

- [November 2023 - December 2023]
- Key Activities: Testing Drone and Recalibration due to different weight.

4. Evaluation and Reporting Phase:

- [December 2023-December 2023]
- Key Activities: Continuous testing , Presentation , Reports , Flowcharts

D. Budget Breakdown:

1. Hardware Costs:

- Motors:- 1600 INR
- Other Sensors: 4000 INR
- Drone Frame:- 800 INR
- ESP 32- 400 INR
- PixHawk Power Module:-
- GPS Module:- 1700 INR
- PPM Encoder- 800 INR

2. Total Project Budget: INR 9300

E. Ethical Considerations:

1. Privacy:

- Describe measures taken to ensure the privacy of individuals captured in visual data.

2. Data Security:

- Explain how data collected from sensors is securely stored and processed to prevent unauthorized access.

3. Regulatory Compliance:

- Detail steps taken to comply with local, national, and international regulations governing drone detection technologies.

4. Informed Consent:

- If applicable, describe the process of obtaining informed consent for data collection, especially in areas involving public surveillance.